

```

NNN      NNN      EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  AAAAAAAAAAA  CCCCCCCCCCCC  PPPPPPPPPPPP
NNN      NNN      EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  AAAAAAAAAAA  CCCCCCCCCCCC  PPPPPPPPPPPP
NNN      NNN      EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  AAAAAAAAAAA  CCCCCCCCCCCC  PPPPPPPPPPPP
NNN      NNN      EEE              TTT              AAA              AAA  CCC              PPP      PPP
NNN      NNN      EEE              TTT              AAA              AAA  CCC              PPP      PPP
NNN      NNN      EEE              TTT              AAA              AAA  CCC              PPP      PPP
NNNNNN   NNN      EEE              TTT              AAA              AAA  CCC              PPP      PPP
NNNNNN   NNN      EEE              TTT              AAA              AAA  CCC              PPP      PPP
NNNNNN   NNN      EEE              TTT              AAA              AAA  CCC              PPP      PPP
NNN      NNN      EEEEEEEEEEEEEEE  TTT              AAA              AAA  CCC              PPPPPPPPPPPP
NNN      NNN      EEEEEEEEEEEEEEE  TTT              AAA              AAA  CCC              PPPPPPPPPPPP
NNN      NNN      EEEEEEEEEEEEEEE  TTT              AAA              AAA  CCC              PPPPPPPPPPPP
NNN      NNN      EEE              TTT              AAA              AAA  CCC              PPP
NNN      NNN      EEE              TTT              AAA              AAA  CCC              PPP
NNN      NNN      EEE              TTT              AAA              AAA  CCC              PPP
NNN      NNN      EEE              TTT              AAA              AAA  CCC              PPP
NNN      NNN      EEE              TTT              AAA              AAA  CCC              PPP
NNN      NNN      EEEEEEEEEEEEEEE  TTT              AAA              AAA  CCCCCCCCCCCC  PPP
NNN      NNN      EEEEEEEEEEEEEEE  TTT              AAA              AAA  CCCCCCCCCCCC  PPP
NNN      NNN      EEEEEEEEEEEEEEE  TTT              AAA              AAA  CCCCCCCCCCCC  PPP

```


NN	NN	EEEEEEEEEE	TTTTTTTTTT	DDDDDDDD	LL	EEEEEEEEEE	
NN	NN	EEEEEEEEEE	TTTTTTTTTT	DDDDDDDD	LL	EEEEEEEEEE	
NN	NN	EE	TT	DD	DD	EE	
NN	NN	EE	TT	DD	DD	EE	
NNNN	NN	EE	TT	DD	DD	EE	
NNNN	NN	EE	TT	DD	DD	EE	
NN	NN	EEEEEEEE	TT	DD	DD	EEEEEEEE	
NN	NN	EEEEEEEE	TT	DD	DD	EEEEEEEE	
NN	NNNN	EE	TT	DD	DD	EE	
NN	NNNN	EE	TT	DD	DD	EE	
NN	NN	EE	TT	DD	DD	EE	
NN	NN	EE	TT	DD	DD	EE	
NN	NN	EEEEEEEEEE	TT	DDDDDDDD	LLLLLLLLLL	EEEEEEEEEE
NN	NN	EEEEEEEEEE	TT	DDDDDDDD	LLLLLLLLLL	EEEEEEEEEE

LL	IIIIII	SSSSSSSS
LL	IIIIII	SSSSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SSSSSS
LL	II	SSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LLLLLLLLLL	IIIIII	SSSSSSSS
LLLLLLLLLL	IIIIII	SSSSSSSS

(2)	72	Declarations
(3)	257	DLE\$DISPATCH - Dispatch newly recieved DLE IRP
(4)	322	DLE\$ACCESS - Handle IO\$ ACCESS function
(5)	434	DLE\$LPD STATUS - Check completion of MOP transition
(6)	525	BC ACCESS - Handle DLE access to broadcast circuit
(7)	583	DLE\$SETMODE - Process IO\$ SETMODE request
(8)	672	DLE\$DEACCESS - Process IO\$ DEACCESS request
(9)	735	LEAVE MOP STATE - Leave MOP state
(10)	777	DLE\$CANCEL - Process DLE cancel request
(11)	799	DLE\$BC_UP - Initialize DLE on broadcast circuit
(12)	884	DLE\$BC_DOWN - Cleanup DLE on broadcast circuit
(13)	951	INIT UNSOL_CHAN - Initialize channel for unsolicited msgs
(14)	1014	ISSUE NI_READ - Issue read request to NI driver
(15)	1104	RCV DLE_MSG - Receive unsolicited DLE message
(16)	1216	DLE\$MOP_REQUEST - Partner has requested MOP mode
(17)	1296	STARTUP_MOM - Start MOM process
(18)	1364	ATTACH UNSOL_MSG - Attach unsolicited message
(19)	1441	DLE\$PRC_EXIT - Handle MOM process termination


```
0000 1 .TITLE NETDLE - NETACP DLE processing
0000 2 .IDENT 'V04-000'
0000 3
0000 4 *****
0000 5 *
0000 6 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 * ALL RIGHTS RESERVED.
0000 9 *
0000 10 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 * TRANSFERRED.
0000 16 *
0000 17 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 * CORPORATION.
0000 20 *
0000 21 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 *
0000 24 *
0000 25 *****
0000 26
0000 27 ++
0000 28 FACILITY: DECnet-VAX
0000 29
0000 30 ABSTRACT:
0000 31
0000 32 This module contains most of the DLE process-level code in
0000 33 NETACP. It works with the DLE driver (NDDRIVER) to implement
0000 34 DLE to allow programs direct access to DECnet circuits. This
0000 35 is primarily used to implement MOP support.
0000 36
0000 37 ENVIRONMENT:
0000 38
0000 39 MODE = KERNEL
0000 40
0000 41 AUTHOR:
0000 42
0000 43 Tim Halvorsen, January 1983
0000 44
0000 45 MODIFIED BY:
0000 46
0000 47 V003 TMH0003 Tim Halvorsen 24-Aug-1984
0000 48 Prevent duplicate MOM processes from being started due
0000 49 to unsolicited messages received AFTER MOM has issued
0000 50 its ACCESS but before it has established a connection
0000 51 with the node (via SETMODE). This is done by simply
0000 52 leaving the unsolicited message which started MOM in the
0000 53 unsolicited queue for the life of the MOM process, causing
0000 54 any new unsolicited messages which "squeak through" to be
0000 55 dropped rather than starting a new MOM process.
0000 56
0000 57 V002 TMH0002 Tim Halvorsen 28-Apr-1983
```


0000 58 :
0000 59 :
0000 60 :
0000 61 :
0000 62 :
0000 63 :
0000 64 :
0000 65 :
0000 66 :
0000 67 :
0000 68 :
0000 69 :
0000 70 :--

V001

Change loopback assistant multicast address to
the one listed in the Ethernet V2 spec.

TMH0001 Tim Halvorsen 29-Mar-1983
Compute a unique MOM process name, so that multiple
service operations can occur on the same circuit.
Fix deallocation of BC blocks to wait for all outstanding
I/O to rundown before deallocating the block.
Add protective code to prevent multiple MOMs from starting
up if the remote station sends requests too often - if we
receive another message while a MOM process is still starting,
it is ignored.


```
0000 72 .SBTTL Declarations
0000 73 ;
0000 74 ; VMS definitions
0000 75 ;
0000 76
0000 77 $ABDDEF ; ACP buffer descriptor
0000 78 $CCBDEF ; Channel control block
0000 79 $CXBDEF ; Complex buffer
0000 80 $ddbDEF ; Device data block
0000 81 $DDTDEF ; Driver dispatch table
0000 82 $DYNDEF ; Structure types
0000 83 $IRPDEF ; I/O request packet
0000 84 $IODEF ; I/O function codes
0000 85 $JIBDEF ; Job information block
0000 86 $PCBDEF ; Process control block
0000 87 $UCBDEF ; Device unit control block
0000 88
0000 89 ;
0000 90 ; Network definitions
0000 91 ;
0000 92
0000 93 $DWBDEF ; DLE window control block
0000 94 $EVCDEF ; Event logging parameter codes
0000 95 $LPDDEF ; Logical path descriptor (circuit)
0000 96 $NETSYMDEF ; Get NET$C_IPL symbol
0000 97 $NFBDEF ; Network parameter codes
0000 98 $NMADEF ; NICE parameter codes
0000 99 $WQEDEF ; Work queue entries
0000 100
0000 101 ;
0000 102 ; Define symbols for timer qualifiers
0000 103 ;
0000 104
00000001 0000 105 TID_C_READSUP = 1 ; NI receive "wait" timer
0000 106
00000004 0000 107 WQESC_QUAL_DLE = 4 ; && temp &&
0000 108
0000 109 ;
0000 110 ; Define format of broadcast circuit "default protocol user" context block.
0000 111 ; This block holds all context related to enabling this process to receive
0000 112 ; all unsolicited messages ("default user") for the MOP protocol types on
0000 113 ; a broadcast circuit, specifically "load/dump" and "loopback" protocol types.
0000 114 ;
0000 115
0000 116 $DEFINI BC GLOBAL ; (GLOBAL is only for debugging)
0000 117
0000 118 $DEF BC_L_FLINK .BLKL 2 ; Forward/backward queue links
0008 119 $DEF BC_W_SIZE .BLKW 1 ; Size of structure
000A 120 $DEF BC_B_TYPE .BLKB 1 ; Type of structure
000B 121 $DEF BC_B_FLAGS .BLKB 1 ; Flags
000C 122 _VIELD BC_O,<- ;
000C 123 -VIELD <DELETE,,M>,- ; Block is marked for deallocation
000C 124 >
0000000E 000C 125 $DEF BC_B_REFCNT .BLKB 1 ; # of IOWQEs still outstanding
000D 126 .BLKB 1 ; (spare for alignment)
000E 127 $DEF BC_W_LPD .BLKW 1 ; LPD ID of broadcast circuit
0010 128 $DEF BC_W_LD_CHAN .BLKW 1 ; Channel for "load/dump" protocol
```



```
0012 129 $DEF BC_W_LP_CHAN .BLKW 1 ; Channel for "loopback" protocol
0014 130 $DEF BC_Q_PND_RCV .BLKL 2 ; Listhead of pending receive IOWQEs
001C 131 $DEF BC_Q_CUR_RCV .BLKL 2 ; Listhead of current receive IOWQEs
0024 132 $DEF BC_Q_UNSQL_MSGS .BLKL 2 ; Listhead for received unsolicited msgs
002C 133 $DEF BC_C_LENGTH ; Length of structure
002C 134
002C 135 $DEFEND BC
0000 136
0000 137 ;
0000 138 ; Define format of an unsolicited message context block
0000 139 ;
0000 140
0000000E 0000 141 NIHDRSIZ = 14 ; Size of NI datalink header
0000 142
0000 143 $DEFINI IOWQE GLOBAL ; (GLOBAL is only for debugging)
0000 144
00000C24 0000 145 . = WQESC_LENGTH ; Start just after standard WQE
0024 146
0024 147 $DEF IOWQE_Q_IOSB .BLKL 2 ; I/O status block
00000026 002C 148 IOWQE_W_MSGLEN = IOWQE_Q_IOSB+2 ; Message length
002C 149 $DEF IOWQE_W_CHAN .BLKW 1 ; Channel to datalink
00000030 002E 150 .BLKW 1 ; (spare for alignment)
0030 151 $DEF IOWQE_L_PID .BLKL 1 ; IPID of MOM process for this msg
0034 152 $DEF IOWQE_L_BC .BLKL 1 ; Address of corresponding BC block
0038 153 $DEF IOWQE_G_NIHDR .BLKB NIHDRSIZ ; NI datalink header
0046 154 $DEF IOWQE_G_MSG .BLKB 1500 ; Actual message (allow for largest)
0622 155 $DEF IOWQE_C_LENGTH
0622 156
0622 157 $DEFEND IOWQE
0000 158
0000 159 ;
0000 160 ; Read/write storage
0000 161 ;
0000 162
00000000 163 .PSECT NET_IMPURE,WRT,NOEXE, LONG
0000 164
00000000' 0000 165 DLE_ACC:
00000000' 0004 166 .ADDRESS DLE_ACC ; Queue of DLE IOS_ACCESS IRPs
0008 167 .ADDRESS DLE_ACC ; waiting for circuit to go into MOP
0008 168
00000008' 0008 169 BC_QUEUE:
00000008' 000C 170 .ADDRESS BC_QUEUE ; Queue of BC blocks for all broadcast
0010 171 .ADDRESS BC_QUEUE ; circuits in the "run" state
00000018 0010 172
0018 173 IOSB: .BLKL 2 ; General purpose I/O status block
00000000 174
00000000 175 .PSECT NET_PURE,NOWRT,NOEXE, LONG
0000 176
0000 177 ;
0000 178 ; Define storage needed to startup MOM
0000 179 ;
0000 180
0000000A 0000 181 MAX_MOM_PROC = 10 ; Maximum number of simultaneous
0000 182 ; MOM processes for a single circuit
0000 183 MOM_OBJ_NAM:
4D 4F 4D 24 00' 0000 184 .ASCIC '$MOM' ; Name of MOM object
04 0000
```



```
4C 55 21 5F 44 41 21 5F 4D 4F 4D 00' 0005 185 MOM_PRCNAM:
OB 0005 186 .ASCII 'MOM;!AD;!UL' ; MOM process name
0011 187
0011 188 :
0011 189 : UNA "setmode" parameters for load/dump protocol
0011 190 :
0011 191
0011 192 LD_PARAMS:
0B0E 0011 193 .WORD NMASC_PCLI_PTY ; Protocol type = 60-01
00000160 0013 194 .LONG ^X0160
0B1E 0017 195 .WORD NMASC_PCLI_ACC ; Protocol access mode = SHARED
00000001 0019 196 .LONG NMASC_ACC_SHR
0AF1 001D 197 .WORD NMASC_PCLI_BUS ; Buffer size = 1498 (2 bytes for PAD)
000005DA 001F 198 .LONG 1498
0451 0023 199 .WORD NMASC_PCLI_BFN ; Number of buffers = 2
00000002 0025 200 .LONG 2
0B0F 0029 201 .WORD NMASC_PCLI_MCA ; Reception of multicast messages:
0008 002B 202 .WORD 8 ; (8 byte string follows)
0001 002D 203 .WORD NMASC_LINMC_SET ; Enable reception of multicast
010000AB 002F 204 .LONG ^X010000AB ; "dump/load assistance"
0000 0033 205 .WORD 0
0B1A 0035 206 .WORD NMASC_PCLI_PAD ; Padding length word = ON
00000000 0037 207 .LONG NMASC_STATE_ON
0B18 003B 208 .WORD NMASC_PCLI_PRM ; Promiscuous mode = OFF
00000001 003D 209 .LONG NMASC_STATE_OFF
0B19 0041 210 .WORD NMASC_PCLI_MLT ; Multicast address state = OFF
00000001 0043 211 .LONG NMASC_STATE_OFF
0B1B 0047 212 .WORD NMASC_PCLI_DCH ; Data chaining = OFF
00000001 0049 213 .LONG NMASC_STATE_OFF ; (DLE driver can't handle multiple CXBs)
0B1C 004D 214 .WORD NMASC_PCLI_CRC ; CRC generation = ON
00000000 004F 215 .LONG NMASC_STATE_ON
0053 216
0053 217 LD_SETMODE:
00000042 0053 218 .LONG .-LD_PARAMS ; Descriptor of above buffer
00000011' 0057 219 .ADDRESS LD_PARAMS
005B 220
005B 221 :
005B 222 : UNA "setmode" parameters for loopback protocol
005B 223 :
005B 224
005B 225 LP_PARAMS:
0B0E 005B 226 .WORD NMASC_PCLI_PTY ; Protocol type = 90-00
00000090 005D 227 .LONG ^X0090
0B1E 0061 228 .WORD NMASC_PCLI_ACC ; Protocol access mode = SHARED
00000001 0063 229 .LONG NMASC_ACC_SHR
0AF1 0067 230 .WORD NMASC_PCLI_BUS ; Buffer size = 1500
000005DC 0069 231 .LONG 1500
0451 006D 232 .WORD NMASC_PCLI_BFN ; Number of buffers = 2
00000002 006F 233 .LONG 2
0B0F 0073 234 .WORD NMASC_PCLI_MCA ; Reception of multicast messages:
0008 0075 235 .WORD 8 ; (8 byte string follows)
0001 0077 236 .WORD NMASC_LINMC_SET ; Enable reception of multicast
000000CF 0079 237 .LONG ^X000000CF ; "loopback assistance"
0000 007D 238 .WORD 0
0B1A 007F 239 .WORD NMASC_PCLI_PAD ; Padding length word = OFF
00000001 0081 240 .LONG NMASC_STATE_OFF
```



```
00000001 0085 241 .WORD NMASC_PCLI_PRM ; Promiscuous mode = OFF
00000001 0087 242 .LONG NMASC_STATE_OFF
00000001 008B 243 .WORD NMASC_PCLI_MLT ; Multicast address state = OFF
00000001 008D 244 .LONG NMASC_STATE_OFF
00000001 0091 245 .WORD NMASC_PCLI_DCH ; Data chaining = OFF
00000001 0093 246 .LONG NMASC_STATE_OFF ; (DLE driver can't handle multiple CXBs)
00000000 0097 247 .WORD NMASC_PCLI_CRC ; CRC generation = ON
00000000 0099 248 .LONG NMASC_STATE_ON
00000000 009D 249
00000042 009D 250 LP_SETMODE:
0000005B' 00A1 251 .LONG .-LP_PARAMS ; Descriptor of above buffer
00000000 00A5 252 .ADDRESS LP_PARAMS
00000000 00A5 253
00000000 00A5 254
00000000 255 .PSECT NET_CODE,NOWRT,EXE
```

```
0000 257 .SBTTL DLE$DISPATCH - Dispatch newly recieved DLE IRP
0000 258 :+
0000 259 : DLE$DISPATCH - Dispatch newly received DLE IRP
0000 260 :
0000 261 : This routine is called from AQB dispatching when an IRP is dequeued
0000 262 : which has the PHYSIO flag set in the IRP flags. This flag is used
0000 263 : by convention between NETDRIVER and NDDRIVER to distinguish between
0000 264 : various flavors of IRPs.
0000 265 :
0000 266 : Inputs:
0000 267 :
0000 268 : R3 = IRP address
0000 269 :
0000 270 : Outputs:
0000 271 :
0000 272 : None - the IRP is always returned to the driver.
0000 273 :
0000 274 DLE$DISPATCH::
0000 275 EXTZV #IRP$V_FCODE,- ; Get function code
0002 276 #IRP$S_FCODE,-
0003 277 IRP$W_FUNC(R3),R7
0006 278 $DISPATCH R7,<-
0006 279 <IO$_ACCESS, 30$>,-
0006 280 <IO$_ACPCONTROL, 40$>,-
0006 281 <IO$_DEACCESS, 50$>,-
0006 282 <IO$_SETMODE, 60$>>
0000'8F 3C 0036 283 10$: MOVZWL #SS$_ILLIOFUNC,- ; Say "illegal I/O function"
38 A3 003A 284 IRP$L_IOST1(R3)
28 11 003C 285 BRB 90$ ; Exit
003E 286 :
003E 287 : ACCESS function - dispatch to connect processor
003E 288 :
0034 30 003E 289 30$: BSBW DLE$ACCESS ; Process IO$_ACCESS function
23 11 0041 290 BRB 90$ ; Exit
0043 291 :
0043 292 : ACP Control
0043 293 :
0043 294 40$: BBS #IRP$V_COMPLX,- ; If normal IO$_ACPCONTROL, then
EE 2A A3 0045 295 IRP$W_STS(R3),10$ ; inform user we don't support them
18 A3 01 8A 0048 296 BICB #1,IRP$L_WIND(R3) ; Clear interlock bit in case an
004C 297 : IO$_ACCESS or IO$_DEACCESS is pending
02E2 30 004C 298 BSBW DLE$CANCEL ; Do cancel-related work
15 11 004F 299 BRB 90$ ; Continue
0051 300 :
0051 301 : DEACCESS function
0051 302 :
0051 303 50$: BICL3 #1,IRP$L_WIND(R3),R6 ; Get DWB without interlock bit
0056 304 BGEQ 90$ ; If GEQ then no DWB
024C 30 0058 305 BSBW DLE$DEACCESS ; Process IO$_DEACCESS function
09 11 005B 306 BRB 90$ ; Continue
005D 307 :
005D 308 : SETMODE function
005D 309 :
005D 310 60$: MOVL IRP$L_WIND(R3),R6 ; Get DWB address
0061 311 BGEQ 90$ ; If GEQ then no DWB
01B9 30 0063 312 BSBW DLE$SETMODE ; Process IO$_SETMODE function
0066 313 :
```


NETDLE
V04-000

- NETACP DLE processing
DLE\$DISPATCH - Dispatch

L 15

16-SEP-1984 01:24:27 VAX/VMS Macro V04-00
5-SEP-1984 02:19:17 [NETACP.SRC]NETDLE.MAR;1

Page 8
(3)

```

      0066 314      ; Give the IRP back to the DLE driver with the I/O status setup
      0066 315      ;
53      DS 0066 316 90$: fSTL R3      ; Did IRP get tucked away somewhere
OA      13 0068 317      ; If so, exit
55      1C A3 D0 006A 318      ; Get UCB address
00000000 GF 16 006E 319      ; Queue packet to driver
      05 0074 320 100$: RSB      ; Done
```

```
0075 322 .SBTTL DLE$ACCESS - Handle IOS_ACCESS function
0075 323 :+
0075 324 : DLE$ACCESS - Process IOS_ACCESS function for a DLE channel
0075 325 :
0075 326 : This routine is entered after the initial IOS_ACCESS processing
0075 327 : done in the DLE driver. It's main function is to perform all
0075 328 : those things which must be done in process context in order to
0075 329 : setup the connection between DLE user and the datalink.
0075 330 :
0075 331 : Inputs:
0075 332 :
0075 333 : R3 = IRP address
0075 334 :
0075 335 : P1 = Circuit name for DLE I/O
0075 336 :
0075 337 : Outputs:
0075 338 :
0075 339 : R3 = IRP address, 0 if not to be returned to driver yet.
0075 340 : IRP$IOST1 = I/O status
0075 341 :
0075 342 DLE$ACCESS:
0075 343 CLRL IRP$I_EXTEND(R3) ; Assume no rcvd msg to be returned
0075 344 :
0075 345 : Construct a descriptor of the circuit name
0075 346 :
0075 347 ADDL3 @IRP$I_SVAPTE(R3),- ; Get address of P1 ABD
0075 348 #ABD$C_FIB*ABD$C_LENGTH,R4
0075 349 MOVZWL ABD$W_COUNT(R4),R7 ; Get length of circuit name
0075 350 MOVZWL ABD$W_TEXT(R4),R1 ; Get offset to circuit name
0075 351 MOVAB 1+ABD$W_TEXT(R4)[R1],R8 ; Get address of text (skip acmode)
0075 352 :
0075 353 : Locate the CRI and LPD for the circuit, and make sure it is
0075 354 : in a state to handle MOP mode.
0075 355 :
0075 356 MOVL NET$GL_CNR_CRI,R11 ; Point to CRI root block
0075 357 CLRL R10 ; Start at beginning of CRI list
0075 358 MOVZWL #SS$_NOSUCHDEV,R0 ; Setup default error code
0075 359 $SEARCH egl_cri,s,nam ; Lookup CRI by circuit name
0075 360 BLBC R0,91$ ; If error detected, then report it
0075 361 $GETFLD cri,l,sta ; Get circuit state
0075 362 MOVZWL #SS$_DEVINACT,R0 ; Assume circuit not on
0075 363 CMPL R8,#NMASC_STATE_OFF ; Circuit off?
0075 364 BEQL 91$ ; If so, report an error
0075 365 BSBW NET$LOCATE_LPD ; Get LPD address
0075 366 BLBC R0,91$ ; Exit if error detected
0075 367 MOVL IRP$I_DIAGBUF(R3),R0 ; Get DWB address
0075 368 MOVW LPD$W_PTH(R6),- ; Store LPD ID of circuit
0075 369 DWB$W_PATH(R0) ; into DLE window block
0075 370 $GETFLD cri,v,ser ; Service functions enabled?
0075 371 MOVZWL #SS$_IVMODE,R0 ; Assume service disabled
0075 372 BLBS R8,9T$ ; If disabled, then report error
0075 373 BBS #LPD$V_X25,- ; No service is allowed
0075 374 LPD$W_STS(R6),91$ ; on X.25 DLM circuits
0075 375 :
0075 376 : If this is a multiaccess circuit, such as Ethernet,
0075 377 : then skip the circuit transition, since there is no
0075 378 : circuit mode.
```

54 A3 D4 0075 343 CLRL IRP\$I_EXTEND(R3) ; Assume no rcvd msg to be returned

2C B3 C1 0075 344 :
54 08 0075 345 : Construct a descriptor of the circuit name
57 02 A4 3C 0075 346 :
51 64 3C 0075 347 ADDL3 @IRP\$I_SVAPTE(R3),- ; Get address of P1 ABD
58 01 A441 9E 0075 348 #ABD\$C_FIB*ABD\$C_LENGTH,R4
0075 349 MOVZWL ABD\$W_COUNT(R4),R7 ; Get length of circuit name
0075 350 MOVZWL ABD\$W_TEXT(R4),R1 ; Get offset to circuit name
0075 351 MOVAB 1+ABD\$W_TEXT(R4)[R1],R8 ; Get address of text (skip acmode)
0075 352 :
0075 353 : Locate the CRI and LPD for the circuit, and make sure it is
0075 354 : in a state to handle MOP mode.
0075 355 :
5B 00000000'EF D0 0075 356 MOVL NET\$GL_CNR_CRI,R11 ; Point to CRI root block
5A D4 0075 357 CLRL R10 ; Start at beginning of CRI list
50 0000'8F 3C 0075 358 MOVZWL #SS\$_NOSUCHDEV,R0 ; Setup default error code
48 50 E9 0075 359 \$SEARCH egl_cri,s,nam ; Lookup CRI by circuit name
0075 360 BLBC R0,91\$; If error detected, then report it
50 0000'8F 3C 0075 361 \$GETFLD cri,l,sta ; Get circuit state
01 58 D1 0075 362 MOVZWL #SS\$_DEVINACT,R0 ; Assume circuit not on
31 13 0075 363 CMPL R8,#NMASC_STATE_OFF ; Circuit off?
FF3D' 30 0075 364 BEQL 91\$; If so, report an error
2B 50 E9 0075 365 BSBW NET\$LOCATE_LPD ; Get LPD address
50 4C A3 D0 0075 366 BLBC R0,91\$; Exit if error detected
20 A6 B0 0075 367 MOVL IRP\$I_DIAGBUF(R3),R0 ; Get DWB address
3E A0 0075 368 MOVW LPD\$W_PTH(R6),- ; Store LPD ID of circuit
0075 369 DWB\$W_PATH(R0) ; into DLE window block
50 0000'8F 3C 0075 370 \$GETFLD cri,v,ser ; Service functions enabled?
0D 58 E8 0075 371 MOVZWL #SS\$_IVMODE,R0 ; Assume service disabled
07 E0 0075 372 BLBS R8,9T\$; If disabled, then report error
08 22 A6 0075 373 BBS #LPD\$V_X25,- ; No service is allowed
0075 374 LPD\$W_STS(R6),91\$; on X.25 DLM circuits
0075 375 :
0075 376 : If this is a multiaccess circuit, such as Ethernet,
0075 377 : then skip the circuit transition, since there is no
0075 378 : circuit mode.


```
06 22 0A E1 00E9 379 ;
00CF 30 00EB 380 BBC #LPDSV_BC,- ; Skip if not broadcast
005E 31 00EE 381 LPDSW_STS(R6),10$
00F1 382 BSBW BC_ACCESS ; Handle broadcast DLE access
00F4 383 91$: BRW 90$ ; Return status to DLE driver
00F4 384 10$:
00F4 385 ;
00F4 386 ; Mark the DLE process as the owner of the circuit. If the
00F4 387 ; circuit is already owned, return an error.
00F4 388 $GETFLD cri,l,owpid ; Get PID of DLE owner
0D 50 E9 0101 389 BLBC R0,20$ ; Branch if not currently owned
0C A3 58 D1 0104 390 CMPL R8,IRP$L_PID(R3) ; Is it already owned by process?
07 13 0108 391 BEQL 20$ ; If so, ok to access
50 0000'8F 3C 010A 392 15$: MOVZWL #SS$_DEVALLOC,R0 ; Report circuit already owned
41 11 010F 393 BRB 90$
03 E2 0111 394 20$: BBSS #LPDSV_ACCESS,- ; Mark circuit accessed for DLE
F4 22 A6 0113 395 LPDSW_STS(R6),15$ ; If already accessed, report error
58 0C A3 D0 0116 396 MOVL IRP$L_PID(R3),R8 ; Get caller's PID
FEE3' 30 011A 397 BSBW CNF$POT_FIELD ; Make process owner of the circuit
02 E2 011D 398 BBSS #LPDSV_DLE,- ; Mark in DLE mode
1D 22 A6 011F 399 LPDSW_STS(R6),30$ ; If already in DLE, skip logging event
0122 400 ;
0122 401 ; Log an event indicating the circuit has been accessed
0122 402 ; locally by a process.
0122 403 ;
55 00000000'EF 9E 0122 404 MOVAB NET$AB_EVT_WQE,R5 ; Get address of common WQE
20 A6 B0 0129 405 MOVW LPDSW_PTH(R6),- ; Set LPD ID into WQE
12 A5 012C 406 WQESW-REQIDT(R5)
0140 8F B0 012E 407 MOVW #EVC$C_DLL_LSC,- ; "locally initiated state change"
1C A5 0132 408 WQESW_EVL_CODE(R5)
03 90 0134 409 MOVW #EVC$C_DLC_POLD_RUNG,- ; Old state = RUNNING
1E A5 0136 410 WQESB_EVL_DT1(R5)
04 90 0138 411 MOVW #EVC$C_DLC_POLD_MAIN,- ; New state = MAINTAINANCE
1F A5 013A 412 WQESB_EVL_DT2(R5)
FEC1' 30 013C 413 BSBW NET$EVT_INTRAW ; Log the event record
013F 414 ;
013F 415 ; Bring the circuit up in 'MOP' state.
50 0000'8F 3C 013F 416 30$: MOVZWL #LEV$C_DLE_ACC,R0 ; Setup DLLTRN event code
FEB9' 30 0144 417 BSBW SET_DLC_EVT ; Queue the request
0147 418 ;
0147 419 ; Wait for the circuit to become ready. When it does, the
0147 420 ; routine DLE$LPD_STATUS will be called.
0147 421 ;
0147 422 ;
00000004'FF 63 0E 0147 423 INSQUE (R3),@DLE_ACC+4 ; Insert IRP onto waiting queue
53 D4 014E 424 CLRL R3 ; Indicate IRP not to be returned
04 11 0150 425 BRB 100$
0152 426 ;
0152 427 ;
0152 428 ; An error has been detected. Return the IRP back to the driver.
0152 429 ;
0152 430 ;
38 A3 50 3C 0152 431 90$: MOVZWL R0,IRP$L_IOST1(R3) ; Pass status back in IRP
05 0156 432 100$: RSB
```



```
0157 434 .SBTTL DLE$LPD_STATUS - Check completion of MOP transition
0157 435 :+
0157 436 : DLE$LPD_STATUS - Check completion of MOP transition
0157 437 :
0157 438 : This routine is called when an LPD has made the transition into MOP
0157 439 : state or if an error has occurred. It is always called by DLLTRN
0157 440 : on circuit transitions if the ACCESS flag is set in the LPD.
0157 441 :
0157 442 : If there is a process waiting to access the circuit, then if the
0157 443 : transition was successful, then that process is allowed to proceed
0157 444 : with the access.
0157 445 :
0157 446 : Inputs:
0157 447 :
0157 448 : R6 = LPD address
0157 449 : R0 = Status of attempted MOP transition of circuit
0157 450 :
0157 451 : Outputs:
0157 452 :
0157 453 : None
0157 454 :
0157 455 : R0-R3,R8-R9 are destroyed.
0157 456 :-
0157 457 DLE$LPD_STATUS::
30 BB 0157 458 PUSH R4,R5 ; Save registers
0159 459 :
0159 460 : Locate the DWB corresponding to the process attempting
0159 461 : the circuit ACCESS.
0159 462 :
51 00000000 EF 9E 0159 463 MOVAB DLE_ACC,R1 ; Get address of DLE ACCESS IRP listhead
53 51 D0 0160 464 MOVL R1,R3 ; Setup for loop
53 63 D0 0163 465 10$: MOVL (R3),R3 ; Skip to next IRP in list
51 53 D1 0166 466 CMPL R3,R1 ; End of list?
54 39 13 0169 467 BEQL 60$ ; If so, then ignore the status
4C A3 D0 016B 468 MOVL IRP$L_DIAGBUF(R3),R4 ; Get DWB address for ACCESS request
20 A6 B1 016F 469 CMPW LPD$W_PTH(R6),- ; Is it for this circuit?
3E A4 0172 470 DWB$W_PATH(R4)
ED 12 0174 471 BNEQ 10$ ; If not, keep looking
53 63 OF 0176 472 REMQUE (R3),R3 ; Remove from pending ACCESS list
10 50 E9 0179 473 BLBC R0,20$ ; Branch if circuit is down
017C 474 :
017C 475 : Setup the datalink channel and UCB address in DWB
017C 476 :
14 A6 B0 017C 477 MOVW LPD$W_CHAN(R6),- ; Save channel to datalink
4C A4 017F 478 DWB$W_DLL_CHAN(R4)
10 A6 D0 0181 479 MOVL LPD$L_UCB(R6),- ; Save UCB of datalink
48 A4 0184 480 DWB$L_DLL_UCB(R4)
0186 481 :
0186 482 : Set the circuit substate to "auto-service"
0186 483 :
27 06 90 0186 484 MOV B #NMASC_LINSS ASE,- ; Set circuit substate
A6 0188 485 LPD$B_SUB_STA(R6)
08 11 018A 486 BRB 50$ ; Pass success back to driver
018C 487 :
018C 488 : Failure to make transition - reset LPD to original state
018C 489 :
50 DD 018C 490 20$: PUSHL R0 ; Save final status
```



```
0167 30 018E 491 BSBW LEAVE_MOP_STATE ; Leave MOP state
50 8ED0 0191 492 POPL R0 ; Restore final status
0194 493 ;
0194 494 ; Report the status back to DLE driver
0194 495 ;
38 A3 50 B0 0194 496 50$: MOVW R0,IRPSL_IOST1(R3) ; Store status in IRP
55 1C A3 D0 0198 497 MOVL IRPSL_UCB(R3),R5 ; Point to the DLE UCB
00000000 GF 16 019C 498 JSB G^EXE$INSIOQ ; Queue packet to DLE driver
19 11 01A2 499 BRB 90$
01A4 500 ;
01A4 501 60$: ;
01A4 502 ; There is no ACCESS request pending for this circuit. If
01A4 503 ; the LPD status is "success", then we can ignore it, since
01A4 504 ; its not relevant except to restart a pending ACCESS.
01A4 505 ;
16 50 E8 01A4 506 BLBS R0,90$ ; Exit if LPD is ok
01A7 507 ;
01A7 508 ; There may be an active DLE session currently in progress
01A7 509 ; over this circuit. Tell the DLE driver to locate all DWBs
01A7 510 ; associated with this circuit, and if any, to abort them.
01A7 511 ;
58 20 A6 3C 01A7 512 MOVZWL LPD$W_PTH(R6),R8 ; Pass path ID to driver
55 00000000 EF D0 01AB 513 MOVL NET$G_DLE_UCB,R5 ; Get DLE UCB address
51 0088 C5 D0 01B2 514 MOVL UCB$D_DDT(R5),R1 ; Get DDT address
04 B1 16 01B7 515 JSB @DDT$C_UNSOINT(R1) ; Call "LPD down" entry point
01BA 516 ; with R0 = status code
01BA 517 ; and R8 = path ID
01BA 518 ;
01BA 519 ; Leave MOP state
01BA 520 ;
013B 30 01BA 521 BSBW LEAVE_MOP_STATE ; Leave MOP state
30 BA 01BD 522 POPR #^M<R4,R5> ; Restore registers
05 01BF 523 RSB
```



```
01C0 525 .SBTTL BC_ACCESS - Handle DLE access to broadcast circuit
01C0 526
01C0 527 :+ BC_ACCESS - Handle DLE access to multiaccess circuit
01C0 528
01C0 529 : This routine is called when an access is being attempted to an
01C0 530 : Ethernet. Since there is no 'MOP mode' for multiaccess circuits,
01C0 531 : we simply assign a new channel to the device, issue a SETMODE to
01C0 532 : enable access to a given destination, and complete the access.
01C0 533
01C0 534 : Inputs:
01C0 535
01C0 536 : R3 = IRP address for ACCESS request
01C0 537 : R6 = LPD address
01C0 538 : R10/R11 = CNF/CNR addresses for CRI
01C0 539
01C0 540 : Outputs:
01C0 541
01C0 542 : R0 = Status code
01C0 543
01C0 544 BC_ACCESS:
01C0 545
01C0 546 : Make sure the circuit is in the "run" state
01C0 547
01C0 548 BBS #LPD$V_RUN,- ; If circuit not ready,
01C2 549 LPD$W_STS(R6),10$
50 08 22 A6 E0 01C5 550 MOVZWL #SS$_DEVINACT,R0 ; Return "circuit not on"
0000'8F 3C 01CA 551 BRW 90$ ; Report the error
0051 31
01CD 552 10$:
01CD 553 : Set a flag in the DWB indicating that this is an NI.
01CD 554
01CD 555 MOVL IRP$L_DIAGBUF(R3),R4 ; Get DWB address
01D1 556 SETBIT #DWB$V_BC,DWB$W_FLAGS(R4) ; Indicate circuit is an NI
01D6 557
01D6 558 : Assign a new channel for this DLE session. Each DLE
01D6 559 : session uses a new NETACP channel so that the demultiplexing
01D6 560 : done by the datalink for received messages (based on the
01D6 561 : source node) can be used by the DLE driver to distinguish
01D6 562 : incoming messages between the various DLE users.
01D6 563
50 0000'8F 3C 01D6 564 MOVZWL #SS$_NOSUCHDEV,R0 ; Setup default error code
01DB 565 $GETFLD cri,$vmsnam ; Get datalink device name
01E8 566 BLBC R0,90$ ; Exit if error detected
7E 57 7D 01EB 567 MOVQ R7,-(SP) ; Push descriptor on stack
50 5E D0 01EE 568 MOVL SP,R0 ; Get address of descriptor
01F1 569 $ASSIGN,S DEVNAM=(R0),- ; Assign a new channel for DLE
01F1 570 CHAN=DWB$W_DLL_CHAN(R4)
5E 08 C0 01FF 571 ADDL #8,SP ; Pop descriptor off stack
19 50 E9 0202 572 BLBC R0,90$ ; Exit if error detected
53 DD 0205 573 PUSHL R3 ; Save IRP address
50 4C A4 3C 0207 574 MOVZWL DWB$W_DLL_CHAN(R4),R0 ; Get channel number
00000000'GF 16 020B 575 JSB G^IOC$VERIFYCHAN ; Get the CCB address; ignore errors
53 8ED0 0211 576 POPL R3 ; Restore IRP address
61 D0 0214 577 MOVL CCB$L_UCB(R1),- ; Save the datalink UCB address
48 A4 0216 578 DWB$L_DLL_UCB(R4)
50 04E5 30 0218 579 BSBW ATTACH_UNSOL MSG ; Pass unsolicited message to user
00' D0 021B 580 MOVL S^#SS$_NORMAL,R0 ; Success
05 021E 581 90$: RSB ; Exit with status
```



```
021F 583 .SBTTL DLE$SETMODE - Process IOS_SETMODE request
021F 584 :+
021F 585 : DLE$SETMODE - Process IOS_SETMODE request at process level
021F 586 :
021F 587 : This routine is called to perform all work needed for the DLE SETMODE
021F 588 : QIO at IPL 0. This includes issuing a SETMODE function to the datalink
021F 589 : driver on the DLE user's behalf. Most of the work done for the SETMODE
021F 590 : has already been accomplished by the DLE driver.
021F 591 :
021F 592 : Inputs:
021F 593 :
021F 594 : R6 = DWB address
021F 595 : R3 = IRP address
021F 596 :
021F 597 : P2 = UNA P2 buffer (used only for DLE access to UNA)
021F 598 : P3 = Ethernet remote address (used only for DLE access to UNA)
021F 599 : P4 = Substate
021F 600 :
021F 601 : Outputs:
021F 602 :
021F 603 : R3 = IRP address, 0 if not to be returned to driver yet.
021F 604 : IRP$L_IOST1 = I/O status
021F 605 :
021F 606 DLE$SETMODE:
021F 607 :
021F 608 : For point-to-point circuits, propagate the (possibly) updated
021F 609 : circuit substate to the LPD (it has already been set in the
021F 610 : DWB by the driver) so that we can see it with existing network
021F 611 : management.
021F 612 :
021F 613 BBS #DWBSV BC,- ; If point-to-point circuit.
021F 614 DWBSW_FLAGS(R6),10$
021F 615 MOVZWL DWBSW_PATH(R6),R8 ; Get LPD ID
021F 616 PUSHL R6 ; Save DWB address
021F 617 BSBW NET$FIND_LPD ; Locate LPD
021F 618 MOVL R6,R2 ; Set LPD address in R2
021F 619 POPL R6 ; Restore DWB address
021F 620 BLBC R0,10$ ; If cannot be found, skip it
021F 621 MOVVB DWBSB_SUBSTA(R6),- ; Copy substate value to LPD
021F 622 LPDSB_SUB_STA(R2)
021F 623 10$:
021F 624 : Construct a descriptor of the P2 buffer (UNA P2 buffer).
021F 625 : If none specified, then skip the SETMODE.
021F 626 :
021F 627 ADDL3 @IRP$L_SVAPTE(R3),- ; Get address of P2 ABD
021F 628 #ABD$C_NAME*ABD$C_LENGTH,R4
021F 629 MOVZWL ABD$W_COUNT(R4),R7 ; Get length of P2
021F 630 BEQL 40$ ; Skip if none
021F 631 MOVZWL ABD$W_TEXT(R4),R1 ; Get offset to P2 data
021F 632 MOVAB 1+ABD$W_TEXT(R4)[R1],R8 ; Get address of P2 data (skip acmode)
021F 633 :
021F 634 : Issue a SETMODE to the datalink driver to establish
021F 635 : "shared" access to the remote node. This allows
021F 636 : more than one DLE user to use the protocol type at the
021F 637 : same time - demultiplexing is done for received messages
021F 638 : based on the remote node address.
021F 639 :
```

17 0E 03 E0
58 3E A6 3C
56 DD
FDD3' 30
52 56 DO
56 8ED0
05 50 E9
46 A6 90
27 A2

2C B3 C1
54 10
57 02 A4 3C
59 13
51 64 3C
58 01 A441 9E


```

      7E 57 7D 024E 640      MOVQ R7,-(SP)      ; Push descriptor of UNA P2 buffer
      50 5E DO 0251 641      MOVL SP,R0         ; Get address of descriptor
52 00000010'EF 9E 0254 642      MOVAB IOSB,R2      ; Get address of I/O status block
      025B 643      $QIOW_S FUNC=#IOS SETMODE!IOSM CTRL!IOSM_STARTUP,- ; Issue request
      025B 644      CHAN=DWB$Q_DLL_CHAN(R6),-
      025B 645      EFN=#NET$C_EFN_WAIT,-
      025B 646      IOSB=(R2),-
      025B 647      P2=R0
      5E 08 CO 0279 648      ADDL #8,SP         ; Pop descriptor off stack
      23 50 E9 027C 649      BLBC R0,90$        ; Exit if error detected
      50 62 3C 027F 650      MOVZWL (R2),R0      ; Get final I/O status
      07 50 E8 0282 651      BLBS R0,30$        ; Exit if ok
3C A3 04 A2 DO 0285 652      MOVL 4(R2),IRP$L_IOST2(R3) ; Return UNA longword to user
      16 11 028A 653      BRB 90$              ; Store primary status and exit
      028C 654 30$:
      028C 655      ;
      028C 656      ; As a result of a SETMODE to the UNA driver for LIMITED protocol
      028C 657      ; access, the UNA driver may have evaporated the UCB we initially
      028C 658      ; got after the $ASSIGN, and "integrated" us into an existing UCB
      028C 659      ; for the first user of the protocol type. As a result, we must
      028C 660      ; re-lookup the datalink UCB address immediately after the SETMODE,
      028C 661      ; and reset our saved value, whether it changed or not.
      53 DD 028C 662      PUSHL R3              ; Save IRP address
      50 4C A6 3C 028E 663      MOVZWL DWB$W_DLL_CHAN(R6),R0 ; Get channel number
00000000'GF 16 0292 664      JSB G^IOC$VERIFYCHAN ; Get the CCB address; ignore errors
      53 8ED0 0298 665      PCPL R3            ; Restore IRP address
      61 DO 029B 666      MOVL CCB$L_UCB(R1),- ; Save the datalink UCB address
      48 A6 029D 667      DWB$L_DLL_UCB(R6)
      50 00' DO 029F 668 40$:      MOVL S^#SS$ NORMAL,R0 ; Successful
38 A3 50 B0 02A2 669 90$:      MOVW R0,IRP$L_IOST1(R3) ; Store status in IRP
      05 02A6 670      RSB                    ; Exit with status
```



```
02A7 672 .SBTTL DLE$DEACCESS - Process IO$_DEACCESS request
02A7 673 :+
02A7 674 : DLE$DEACCESS - Process IO$_DEACCESS request
02A7 675 :
02A7 676 : This routine is called to perform all work needed for the DLE DEACCESS
02A7 677 : QIO at IPL 0. If this is a point-to-point circuit, then we must cause
02A7 678 : the circuit to revert back into its original state.
02A7 679 :
02A7 680 : Inputs:
02A7 681 :
02A7 682 :     R6 = DWB address
02A7 683 :     R3 = IRP address
02A7 684 :
02A7 685 : Outputs:
02A7 686 :
02A7 687 :     R3 = IRP address, 0 if not to be returned to driver yet.
02A7 688 :     IRP$L_IOST1 = I/O status
02A7 689 : -
02A7 690 DLE$DEACCESS:
02A7 691 :
02A7 692 :     Locate the circuit data structures based on the LPD ID
02A7 693 :     stored in the DWB at access time.
02A7 694 :
02A7 695 :     MOVL R6,R4 ; Save DWB address for later
58 54 56 D0 02AA 696 :     MOVZWL DWB$W_PATH(R6),R8 ; Get LPD ID
3E A6 3C 02AE 697 :     BEQL 70$ ; If none, report error
41 13 02B0 698 :     BSBW NET$GET_LPD_CRI ; Get LPD, CRI addresses
FD4D' 30 02B3 699 :     BLBC R0,90$ ; Exit if error detected
36 50 E9 02B6 700 :
02B6 701 :     If this is a multiaccess circuit, such as Ethernet,
02B6 702 :     then skip the circuit transition, since there is no
02B6 703 :     circuit "mode".
02B6 704 :
02B6 705 :     BBC #LPD$V_BC,- ; Skip if not broadcast
OD 22 0A E1 02B8 706 :     LPD$W_STS(R6),20$
24 11 02BB 707 :     $DASSGN_S_CHAN=DWB$W_DLL_CHAN(R4) ; Deassign channel to datalink
02C6 708 :     BRB 90$ ; Exit with status
02C8 709 20$:
02C8 710 :     Make sure this user is actually the current "owner"
02C8 711 :     of the circuit.
02C8 712 :
02C8 713 :     $GETFLD cri,l,owpid ; Get the owner PID
19 50 E9 02D5 714 :     BLBC R0,70$ ; If none at all, report an error
OC A3 58 D1 02D8 715 :     CML R8,IRP$L_PID(R3) ; Check if this user is owner
13 12 02DC 716 :     BNEQ 70$ ; If not, return an error
02DE 717 :
02DE 718 :     Leave MOP state
02DE 719 :
02DE 720 :     BSBW LEAVE_MOP_STATE ; Leave MOP state
0017 30 02E1 721 :
02E1 722 :     Bring the circuit down, which will cause it to attempt
02E1 723 :     to re-initialize, this time in normal mode (because the
02E1 724 :     DLE flag is off).
02E1 725 :
02E1 726 :     MOVZWL #LEV$C_LIN_DOWN,R0 ; Setup DLLTRN event code
50 0000'8F 3C 02E6 727 :     BSBW SET_DLE_EVT ; Queue the request
FD17' 30 02E9 728 :     MOVL S^#SS$_NORMAL,R0 ; Success
50 00' D0
```


NETDLE
V04-000

- NETACP DLE processing
DLE\$DEACCESS - Process

H 16

16-SEP-1984 01:24:27
5-SEP-1984 02:19:17

VAX/VMS Macro V04-00
[NETACP.SRC]NETDLE.MAR;1

Page 17
(8)

38	A3	50	B0	02EC	729	90\$:	MOVW	R0,IRP\$L_IOST1(R3)	; Store status in IRP
			05	02F0	730		RSB		; Exit with status
				02F1	731				
50	0000'8F	3C	02F1	732	70\$:	MOVZWL	#SS\$_FILNOTACC,R0		; Circuit not accessed
	F4	11	02F6	733		BRB	90\$		


```
02F8 735 .SBTTL LEAVE_MOP_STATE - Leave MOP state
02F8 736 :+
02F8 737 : LEAVE_MOP_STATE - Leave MOP state for an LPD
02F8 738 :
02F8 739 : This routine is called to reset LPD fields when leaving MOP state.
02F8 740 :
02F8 741 : Inputs:
02F8 742 :
02F8 743 : R10/R11 = CRI pointers
02F8 744 : R6 = LPD address
02F8 745 :
02F8 746 : Outputs:
02F8 747 :
02F8 748 : None
02F8 749 :-
02F8 750 LEAVE_MOP_STATE:
02F8 751 :
02F8 752 : Mark the circuit no longer accessed
02F8 753 :
02F8 754 CLRBIT #LPDSV_ACCESS,- ; Mark no longer accessed
02F8 755 LPDSW_STS(R6)
02FD 756 $CLRFLD cri,l,owpid ; Clear the owner PID
030A 757 :
030A 758 : If we are just leaving MOP mode, then reset circuit
030A 759 : substate and log an event record.
030A 760 :
21 22 02 E5 030A 761 BBCC #LPDSV_DLE,- ; Clear DLE flag
27 A6 90 030C 762 LPDSW_STS(R6),30$ ; If already cleared, skip following
27 A6 90 030F 763 MOV B #NMASC LINSS SYN,- ; Enter "synchronizing" substate
55 00000000'EF 9E 0311 764 LPDSB SUB STA(R6)
20 A6 B0 0313 765 MOVAB NET$AB EVT WQE,R5 ; Get address of common WQE
12 A5 B0 031A 766 MOVW LPDSW_PTH(R6),- ; Set LPD ID into WQE
0140 8F B0 031D 767 WQESW_REQIDT(R5)
1C A5 B0 031F 768 MOVW #EVCSC_DLL_LSC,- ; "locally initiated state change"
04 90 0323 769 WQESW_EVL_CODE(R5)
1E A5 90 0325 770 MOV B #EVCSC_DLC_POLD_MAIN,- ; Old state = MAINTAINANCE
03 90 0327 771 WQESB_EVL_DT1(R5)
1F A5 90 0329 772 MOV B #EVCSC_DLC_POLD_RUNG,- ; New state = RUNNING
FCDO' 30 032B 773 WQESB_EVL_DT2(R5)
05 032D 774 BSBW NET$EVT_INTRAW ; Log the event record
0330 775 30$: RSB
```

```
0331 777 .SBTTL DLE$CANCEL - Process DLE cancel request
0331 778 ;+
0331 779 ; DLE$CANCEL - Process DLE cancel request
0331 780 ;
0331 781 ; This routine is called to perform all work needed for a cancel of a
0331 782 ; DLE "accessed" channel at IPL 0. Presently, nothing needs to be done
0331 783 ; except the datalink cancel I/O already done by the driver.
0331 784 ;
0331 785 ; Inputs:
0331 786 ;
0331 787 ; R3 = IRP address
0331 788 ;
0331 789 ; Outputs:
0331 790 ;
0331 791 ; R3 = IRP address, 0 if not to be returned to driver yet.
0331 792 ; IRP$L_IOST1 = I/O status
0331 793 ; -
0331 794 DLE$CANCEL:
38 50 00' D0 0331 795 MOVL S^#SS$ NORMAL,R0 ; Successful
A3 50 B0 0334 796 MOVW R0,IRP$L_IOST1(R3) ; Store status in IRP
05 0338 797 RSB
```



```
0339 799 .SBTTL DLE$BC_UP - Initialize DLE on broadcast circuit
0339 800 :+
0339 801 : DLE$BC_UP - Initialize DLE on a broadcast circuit which has just come up
0339 802 :
0339 803 : This routine is called when a broadcast circuit has just come up and
0339 804 : entered the "run" state. It sets up NETACP as the "shared" protocol user
0339 805 : of the "load/dump" and "loopback" NI protocols, so that DECnet can
0339 806 : receive requests from other nodes on the NI.
0339 807 :
0339 808 : Inputs:
0339 809 :
0339 810 : R11 = CRI CNR address
0339 811 : R10 = CRI CNF address
0339 812 : R7 = ADJ address
0339 813 : R6 = LPD address
0339 814 : R4 = RCB address
0339 815 :
0339 816 : Outputs:
0339 817 :
0339 818 : R0 = Status code
0339 819 :
0339 820 : R1 is destroyed.
0339 821 :
0339 822 DLE$BC_UP::
0339 823 PUSHRR #M<R2,R3,R4,R5,R6,R7,R8,R9> ; Save registers
0339 824 :
0339 825 : If service functions are disabled for this circuit, then do
0339 826 : not enable "load/dump" or "loopback" protocol types.
0339 827 :
0339 828 $GETFLD cri,l,ser ; Get SERVICE flag
0339 829 BLBS R8,90$ ; Branch if disabled
0339 830 :
0339 831 : Allocate and initialize a new BC context block
0339 832 :
0339 833 MOVZWL #BC_C_LENGTH,R1 ; Size of structure
0339 834 JSB NET$ACLOCATE ; Allocate the block
0339 835 BLBC R0,100$ ; Exit if error detected
0339 836 PUSHLL R2 ; Save address of block
0339 837 MOVCS #0,(SP),#0,#BC_C_LENGTH-12,12(R2) ; Zero the block
0339 838 POPL R5 ; Set R5 to block address
0339 839 MOVAB BC_Q_UNSOL_MSGS(R5),R0 ; Get address of listhead
0339 840 MOVL R0,(R0) ; Init listhead
0339 841 MOVAL (R0)+,(R0)
0339 842 MOVAB BC_Q_PND_RCV(R5),R0 ; Get address of listhead
0339 843 MOVL R0,(R0) ; Init listhead
0339 844 MOVAL (R0)+,(R0)
0339 845 MOVAB BC_Q_CUR_RCV(R5),R0 ; Get address of listhead
0339 846 MOVL R0,(R0) ; Init listhead
0339 847 MOVAL (R0)+,(R0)
0339 848 MOVW LPD$W_PTH(R6),BC_W_LPD(R5) ; Save LPD of associated circuit
0339 849 INSQUE (R5),BC_QUEUE+4 ; Insert block into queue
0339 850 :
0339 851 : Initialize ourselves as the "default user" of the "load/dump"
0339 852 : protocol type.
0339 853 :
0339 854 MOVAB BC_W_LD_CHAN(R5),R3 ; Point to word to receive channel #
0339 855 MOVAB LD_SETMODE,R4 ; Point to descriptor of SETMODE buffer
```

03FC 8F BB 0339 823
033D 824
033D 825
033D 826
033D 827
033D 828
64 58 E8 034A 829
034D 830
034D 831
034D 832
51 2C 3C 034D 833
00000000'EF 16 0350 834
5D 50 E9 0356 835
52 DD 0359 836
OC A2 20 00 6E 00 2C 035B 837
55 8ED0 0362 838
50 24 A5 9E 0365 839
60 50 D0 0369 840
60 80 DE 036C 841
50 14 A5 9E 036F 842
60 50 D0 0373 843
60 80 DE 0376 844
50 1C A5 9E 0379 845
60 50 D0 037D 846
60 80 DE 0380 847
OE A5 20 A6 B0 0383 848
0000000C'FF 65 OE 0388 849
038F 850
038F 851
038F 852
038F 853
53 10 A5 3E 038F 854
54 00000053'EF 9E 0393 855

```
00A5 30 039A 856 BSBW INIT_UN SOL_CHAN ; Initialize channel
16 50 F9 039D 857 BLBC R0,100$ ; Exit if error detected
      03A0 858
      03A0 859 ; Initialize ourselves as the "default user" of the "loopback"
      03A0 860 ; protocol type.
      03A0 861
54 53 12 A5 3E 03A0 862 MOVAV BC_W_LP_CHAN(R5),R3 ; Point to word to receive channel #
0000009D'EF 9E 03A4 863 MOVAB LP_SETMODE,R4 ; Point to descriptor of SETMODE buffer
      0094 30 03AB 864 BSBW INIT_UN SOL_CHAN ; Initialize channel
      05 50 E9 03AE 865 BLBC R0,100$ ; Branch if error detected
03FC 8F BA 03B1 866 90$: POPR #M<R2,R3,R4,R5,R6,R7,R8,R9> ; Restore registers
      05 03B5 867 RSB ; Exit with status
      03B6 868
      03B6 869
      03B6 870 ; An error occurred trying to setup the circuit for service functions.
      03B6 871 ; Log an error, and bring down the circuit.
      03B6 872
      03B6 873
55 00000000'EF 9E 03B6 874 100$: MOVAB NET$AB_EVT_WQE,R5 ; Get address of common WQE
      07 B0 03BD 875 MOVW #EVC$C_NMA_ABS,- ; "aborted service request"
      1C A5 03BF 876 WQESW EVL_CODE(R5)
      04 90 03C1 877 MOVAB #EVC$C_NMA_PRSN_LOE,- ; Reason = "line open error"
      1E A5 03C3 878 WQESB EVL_DT1(R5)
      FC38' 30 03C5 879 BSBW NET$EVT_INTRAW ; Log the event record
50 0000'8F 3C 03C8 880 MOVZWL #LEV$C_CIN_DOWN,R0 ; Setup "circuit down" event
      FC30' 30 03CD 881 BSBW SET_DLE_EVT ; Queue event to DLLTRN
      DF 11 03D0 882 BRB 90$ ; Exit
```



```
03D2 884 .SBTTL DLE$BC_DOWN - Cleanup DLE on broadcast circuit
03D2 885 :+
03D2 886 : DLE$BC_DOWN - Cleanup DLE on broadcast circuit
03D2 887 :
03D2 888 : This routine is called when a broadcast circuit leaves the 'run' state.
03D2 889 : We must deallocate any BC context blocks if they were associated with this
03D2 890 : circuit.
03D2 891 :
03D2 892 : Inputs:
03D2 893 :
03D2 894 : R6 = LPD address
03D2 895 :
03D2 896 : Outputs:
03D2 897 :
03D2 898 : None
03D2 899 :
03D2 900 DLE$BC_DOWN::
3C BB 03D2 901 PUSH R #^M<R2,R3,R4,R5> ; Save registers
03D4 902 :
03D4 903 : Locate the BC block associated with this circuit.
03D4 904 :
51 00000008'EF 9E 03D4 905 MOVAB BC_QUEUE,R1 ; Get address of BC queue
55 51 D0 03DB 906 MOVL R1,R5 ; Setup for loop
55 65 D0 03DE 907 10$: MOVL (R5),R5 ; Skip to next block in queue
51 55 D1 03E1 908 CMPL R5,R1 ; End of list?
OE 59 13 03E4 909 BEQL 90$ ; If not found, skip it
20 A5 B1 03E6 910 CMPW BC_W_LPD(R5),- ; Does the LPD ID match?
20 A6 12 03E9 911 LPD$Q_PTH(R6)
55 65 0F 03EB 912 BNEQ 10$ ; If not, keep looking
03ED 913 REMQUE (R5),R5 ; Remove BC from list
03F0 914 :
03F0 915 : For any non-zero channels, deassign them
03F0 916 :
50 10 A5 3C 03F0 917 MOVZWL BC_W_LD_CHAN(R5),R0 ; Get "load/dump" channel
OA 13 03F4 918 BEQL 20$ ; If nonzero,
03F6 919 $DASSGN_S CHAN=R0 ; Deassign it
50 12 A5 3C 0400 920 20$: MOVZWL BC_W_LP_CHAN(R5),R0 ; Get "loopback" channel
OA 13 0404 921 BEQL 30$ ; If nonzero,
0406 922 $DASSGN_S CHAN=R0 ; Deassign it
0410 923 30$:
0410 924 : Deallocate all unsolicited messages still waiting for
0410 925 : the process to deal with them.
0410 926 :
50 24 B5 0F 0410 927 40$: REMQUE @BC_Q_UN SOL_MSGS(R5),R0 ; Get next unsolicited message
08 1D 0414 928 BVS 45$ ; Branch if none left in queue
00000000'EF 16 0416 929 JSB NET$DEALLOCATE ; Deallocate the block
F2 11 041C 930 BRB 40$ ; Empty the entire queue
041E 931 45$:
041E 932 : Deallocate all receive IOWQEs waiting to be issued to
041E 933 : the NI driver.
041E 934 :
50 14 B5 0F 041E 935 60$: REMQUE @BC_Q_PND_RCV(R5),R0 ; Get next waiting receive IOWQE
08 1D 0422 936 BVS 65$ ; Branch if none left in queue
00000000'EF 16 0424 937 JSB NET$DEALLOCATE ; Deallocate the block
F2 11 042A 938 BRB 60$ ; Empty the entire queue
042C 939 65$:
042C 940 : Deallocate the BC context block
```


			042C	941					
			042C	942					
0C	A5	95	0431	943		SETBIT	#BC_V_DELETE,BC_B_FLAGS(R5)	; Mark block for deletion	
	09	12	0434	944		TSTB	BC_B_REFCNT(R5)	; Are there still receives outstanding?	
			0436	945		BNEQ	90\$; If so, wait for them to complete	
			0436	946				; before deallocating B block	
50	55	D0	0436	946		MOVL	R5,R0	; Set the block address	
00000000	EF	16	0439	947		JSB	NET\$DEALLOCATE	; Deallocate it	
	3C	BA	043F	948	90\$:	POPR	#^M<R2,R3,R4,R5>	; Restore registers	
		05	0441	949		RSB			


```
0442 951 .SBTTL INIT_UN SOL_CHAN - Initialize channel for unsolicited msgs
0442 952
0442 953 :+ INIT_UN SOL_CHAN - Initialize channel for unsolicited messages for a protocol
0442 954
0442 955 : This routine is called to assign a new datalink channel, setup the channel
0442 956 : to be the "default user" of the protocol, so that messages not directly
0442 957 : intended for any other "limited users" of the protocol come to us, and then
0442 958 : issue an asynchronous receive on the channel.
0442 959
0442 960 : Inputs:
0442 961
0442 962 : R10/R11 = CRI pointers
0442 963 : R3 = Address of word to store channel number
0442 964 : R4 = Address of SETMODE P2 buffer
0442 965 : R5 = Address of BC context block
0442 966
0442 967 : Outputs:
0442 968
0442 969 : R0 = Status code
0442 970
0442 971 INIT_UN SOL_CHAN:
0442 972 MOVZWL #SS$, NOSUCHDEV, R0 ; Setup default error status
0442 973 $GETFLD cri, s, vmsnam ; Get datalink device name
0442 974 BLBC R0, 90$ ; Branch if error detected
0442 975 MOVQ R7, -(SP) ; Push descriptor on stack
0442 976 MOVL SP, R0 ; Get address of descriptor
0442 977 $ASSIGN, S, DEVNAM=(R0), - ; Assign channel to NI driver
0442 978 : CHAN=(R3)
0442 979 ADDL #8, SP ; Pop descriptor off stack
0442 980 BLBC R0, 90$ ; Branch if error detected
0442 981
0442 982 : Issue a SETMODE request to the NI driver to establish the
0442 983 : channels as accessing the protocol type as "default user".
0442 984
0442 985 $QIOW, S, FUNC=#IOS$, SETMODE!IOSM_CTRL!IOSM_STARTUP, -
0442 986 : CHAN=(R3), -
0442 987 : EFN=#NETSC$, EFN_WAIT, -
0442 988 : IOSB=IOSB, -
0442 989 : P2=R4
0442 990 BLBC R0, 90$ ; Branch if error detected
0442 991 MOVZWL IOSB, R0 ; Get final I/O status
0442 992 BLBC R0, 90$ ; Branch if error detected
0442 993
0442 994 : Allocate and initialize an IOWQE to to be used to receive
0442 995 : unsolicited messages for this protocol.
0442 996
0442 997 MOVZWL #IOWQE_C_LENGTH-WQESC_LENGTH, R1 ; Get additional storage size
0442 998 MOVL #WQESC_SUB_AST, R0 ; Indicate WQE sub-type
0442 999 BSBW WQES$ALLOCATE ; Allocate a WQE - always succeeds
0442 1000 MOVW (R3), IOWQE_W_CHAN(R2) ; Store channel to datalink
0442 1001 MOVL R5, IOWQE_L_BC(R2) ; Store backpointer to BC block
0442 1002 MOVW BC_W_LPD(R5), - ; Use LPD ID as REQIDT
0442 1003 WQES$ REQIDT(R2)
0442 1004 INSQUE (R2), BCB_Q_PND_RCV+4(R5); Insert on pending receive queue
0442 1005
0442 1006 : Issue asynchronous read on the channel, so that we are
0442 1007 : notified when someone sends us an unsolicited message.
```

50 0000'8F 3C 0442 972
6C 50 E9 0447 973
7E 57 7D 0454 974
50 5E D0 0457 975
045A 976
045D 977
045D 978
5E 08 C0 046A 979
53 50 E9 046D 980
0470 981
0470 982
0470 983
0470 984
0470 985
0470 986
0470 987
0470 988
0470 989
2F 50 E9 0491 990
00000010'EF 3C 0494 991
25 50 E9 049B 992
049E 993
049E 994
049E 995
049E 996
51 05FE 8F 3C 049E 997
50 03 D0 04A3 998
FB57' 30 04A6 999
2C A2 63 B0 04A9 1000
34 A2 55 D0 04AD 1001
0E A5 B0 04B1 1002
12 A2 04B4 1003
18 B5 62 0E 04B6 1004
04BA 1005
04BA 1006
04BA 1007

```

D 1
- NETACP DLE processing 16-SEP-1984 01:24:27 VAX/VMS Macro V04-00
INIT_UNSQL_CHAN - Initialize channel for 5-SEP-1984 02:19:17 [NETACP.SRC]NETDLE.MAR;1

```

Page 25
(13)

51	OE	A5	3C	04BA	1008	:		
		04	10	04BA	1009	MOVZWL	BC W LPD(R5),R1	; Get LPD index
				04BE	1010	BSBB	ISSUE NI_READ	; Issue read request
50	00		D0	04C0	1011	MOVL	S^#SS\$_NORMAL,R0	; Success
			05	04C3	1012	RSB		; Exit with status
					90\$:			

NE Sy
\$\$
\$\$
AB
AB
AB
AB
AC
AC
AC
AC
AT
BC
BC
BC
BC
BC
BC
BC
BC
BC
BC
BC
BC
BC
BI
CC
CN
CN
CN
CN
CN
CN
CX
CX
CX
CX
DD
DL
DL
DL
DL
DL
DL
DL
DL
DL


```
04C4 1014 .SBTTL ISSUE_NI_READ - Issue read request to NI driver
04C4 1015 :+
04C4 1016 : ISSUE_NI_READ - Issue read request to NI driver
04C4 1017 :
04C4 1018 : This routine is called to issue the read request, and return as soon
04C4 1019 : as the request has been queued. All read requests are automatically
04C4 1020 : delayed by 1 second, so that if there is an abnormal node continuously
04C4 1021 : sending messages, we won't get swamped (the NI driver will drop them
04C4 1022 : for us). The delay doesn't affect normal reception, because the NI
04C4 1023 : driver buffers any incoming messages for us, up to a limit.
04C4 1024 :
04C4 1025 : Inputs:
04C4 1026 :
04C4 1027 : R1 = LPD ID for circuit
04C4 1028 :
04C4 1029 : Outputs:
04C4 1030 :
04C4 1031 : None
04C4 1032 :
04C4 1033 : R0-R1 are destroyed.
04C4 1034 :
04C4 1035 : -
04C4 1036 : ISSUE_NI_READ:
04C4 1037 : PUSH R2,R3 ; Save registers
04C4 1038 : ASHL #16,R1,R1 ; Shift LPD ID into upper word
04C4 1039 : MOVW #<<WQE$C_QUAL_DLE>>@8>!,R1 ; Overlay QUAL and EVT fields
04C4 1040 : TID C_READSUP,R1
04C4 1041 : MOVAB B^50$,R2 ; Set address of action routine
04C4 1042 : MOVQ #1*10*1000*1000,R3 ; Wait 1 second
04C4 1043 : BSBW WQE$RESET_TIM ; Wait for timer to fire
04C4 1044 : POPR #M<R2,R3> ; Restore registers
04C4 1045 : RSB
04C4 1046 :
04C4 1047 : : Call here when timer fires
04C4 1048 : 50$: MOVZWL WQE$W_REQIDT(R5),R8 ; Get LPD ID
04C4 1049 : MOVL R5,R0 ; Get timer WQE address
04C4 1050 : BSBW WQE$DEALLOCATE ; Deallocate timer WQE
04C4 1051 :
04C4 1052 : : Locate the BC block associated with this circuit. When found,
04C4 1053 : : if there are any IOWQEs (receives) waiting to be issued to the
04C4 1054 : : NI driver, issue them now.
04C4 1055 :
04C4 1056 : MOVAB BC_QUEUE,R4 ; Get address of BC queue
04C4 1057 : MOVL R4,R5 ; Setup for loop
04C4 1058 : 5$: MOVL (R5),R5 ; Skip to next block in queue
04C4 1059 : CMPL R5,R4 ; End of list?
04C4 1060 : BEQL 90$ ; If not found, skip it
04C4 1061 : CMPW BC_W_LPD(R5),R8 ; Does the LPD ID match?
04C4 1062 : BNEQ 5$ ; If not, keep looking
04C4 1063 : 10$: REMQUE @BC_Q_PND_RCV(R5),R2 ; Get any receives waiting to be issued
04C4 1064 : BVS 5$ ; If none, keep looking
04C4 1065 : INSQUE (R2),@BC_Q_CUR_RCV+4(R5) ; Insert on outstanding receive queue
04C4 1066 : INCB BC_B_REFCT(R5) ; Increment reference count
04C4 1067 : MOVAB IOWQE_G_NIHDR(R2),R0 ; Get address of NI header buffer
04C4 1068 : $QIO_S FUNC=#IOS_READVBLK,- ; Wait for a message to come in
04C4 1069 : CHAN=IOWQE_W_CHAN(R2),-
04C4 1070 : EFN=#NET$C_EFN_ASYN,-
```

53 00000000 52 E4'AF 9E 04CF 1040 MOVAB B^50\$,R2 ; Set address of action routine
51 51 10 78 04C6 1037 ASHL #16,R1,R1 ; Shift LPD ID into upper word
51 0401 8F B0 04CA 1038 MOVW #<<WQE\$C_QUAL_DLE>>@8>!,R1 ; Overlay QUAL and EVT fields
52 E4'AF 9E 04CF 1040 MOVAB B^50\$,R2 ; Set address of action routine
00989680 8F 7D 04D3 1041 MOVQ #1*10*1000*1000,R3 ; Wait 1 second
FB1F' 30 04DE 1042 BSBW WQE\$RESET_TIM ; Wait for timer to fire
OC BA 04E1 1043 POPR #M<R2,R3> ; Restore registers
05 04E3 1044 RSB
58 12 A5 3C 04E4 1048 50\$: MOVZWL WQE\$W_REQIDT(R5),R8 ; Get LPD ID
50 55 DO 04E8 1049 MOVL R5,R0 ; Get timer WQE address
FB12' 30 04EB 1050 BSBW WQE\$DEALLOCATE ; Deallocate timer WQE
54 00000008'EF 9E 04EE 1056 MOVAB BC_QUEUE,R4 ; Get address of BC queue
55 54 DO 04F5 1057 MOVL R4,R5 ; Setup for loop
55 65 DO 04F8 1058 5\$: MOVL (R5),R5 ; Skip to next block in queue
54 55 D1 04FB 1059 CMPL R5,R4 ; End of list?
4D 13 04FE 1060 BEQL 90\$; If not found, skip it
58 0E A5 B1 0500 1061 CMPW BC_W_LPD(R5),R8 ; Does the LPD ID match?
F2 12 0504 1062 BNEQ 5\$; If not, keep looking
52 14 B5 OF 0506 1063 10\$: REMQUE @BC_Q_PND_RCV(R5),R2 ; Get any receives waiting to be issued
EC 1D 050A 1064 BVS 5\$; If none, keep looking
20 B5 62 OE 050C 1065 INSQUE (R2),@BC_Q_CUR_RCV+4(R5) ; Insert on outstanding receive queue
OC A5 96 0510 1066 INCB BC_B_REFCT(R5) ; Increment reference count
50 38 A2 9E 0513 1067 MOVAB IOWQE_G_NIHDR(R2),R0 ; Get address of NI header buffer
0517 1068 \$QIO_S FUNC=#IOS_READVBLK,- ; Wait for a message to come in
0517 1069 CHAN=IOWQE_W_CHAN(R2),-
0517 1070 EFN=#NET\$C_EFN_ASYN,-

```
0517 1071      IOSB=IOWQE_Q_IOSB(R2),-
0517 1072      ASTADR=B^RCV_DLE_MSG_AST,-
0517 1073      ASTPRM=R2,-
0517 1074      P1=IOWQE_G_MSG(R2),-      ; Address of receive buffer
0517 1075      P2=#1500,-      ; Length of receive buffer
0517 1076      P5=R0      ; Address of buffer to receive NI header
24 A2 C5 50 E8 053E 1077      BLBS R0,10$      ; Branch if ok
50      50 3C 0541 1078      MOVZWL R0,IOWQE_Q_IOSB(R2)      ; Store QIO status in IOSB
52      52 DD 0545 1079      PUSHL R2      ; IOWQE address
4E'AF 01 FB 0547 1080      CALLS #1,B^RCV_DLE_MSG_AST      ; Call AST routine
B9      11 054B 1081      BRB 10$      ; Keep scanning
05      05 054D 1082 90$: RSB
054E 1083
054E 1084      ; Receive AST
054E 1085
054E 1086
054E 1087
054E 1088 RCV_DLE_MSG_AST:
0000 054E 1089      .WORD 0
50 04 AC D0 0550 1090      MOVL 4(AP),R0      ; Get WQE address
0554 1091
0554 1092      ; Remove from outstanding receive queue
0554 1093
50 60 OF 0554 1094      REMQUE (R0),R0      ; Remove from queue
0557 1095
0557 1096      ; Queue a work queue entry to process the I/O completion
0557 1097
0557 1098
60'AF 9E 0557 1099      MOVAB B^RCV_DLE_MSG,-      ; Set address of work routine
OC AO 055A 1100      WQESL ACTION(R0)
FAA1' 30 055C 1101      BSBW WQESINSQUE      ; Insert onto work queue
04 055F 1102      RET
```



```
0560 1104 .SBTTL RCV_DLE_MSG - Receive unsolicited DLE message
0560 1105
0560 1106 :+ RCV_DLE_MSG - Receive unsolicited DLE message
0560 1107
0560 1108 : This routine is called when a receive completes on one of the DLE "shared"
0560 1109 : channels. This means that an unsolicited message has come in which could
0560 1110 : not be associated with any existing protocol user. Our action is to start
0560 1111 : up a MOM process to handle the DLE session.
0560 1112
0560 1113 : Inputs:
0560 1114
0560 1115 : R5 = IOWQE address
0560 1116
0560 1117 : Outputs:
0560 1118
0560 1119 : None
0560 1120
0560 1121 RCV_DLE_MSG:
54 34 A5 D0 0560 1122 MOVL IOWQE_L BC(R5),R4 ; Get BC address
   OC A4 97 0564 1123 DECB BC_B_REFCNT(R4) ; Decrement outstanding I/O count
0567 1124
0567 1125 : Locate the CRI associated with this circuit
0567 1126
58 0E A4 3C 0567 1127 MOVZWL BC_W_LPD(R4),R8 ; Get LPD ID
   FA92' 30 056B 1128 BSBW NET$GET_LPD_CRI ; Get LPD, CRI addresses
   2C 50 E9 056E 1129 BLBC R0,5$ ; Exit if error detected
0571 1130
0571 1131 : If the BC is marked for rundown, then this I/O completion
0571 1132 : should be ignored, and the BC deallocated if possible.
0571 1133
10 0B A4 00 E1 0571 1134 BBC #BC V DELETE,BC_B_FLAGS(R4),4$ ; If BC marked for rundown,
   OC A4 95 0576 1135 TSTB BC_B_REFCNT(R4) ; Any more receives still outstanding?
   22 12 0579 1136 BNEQ 5$ ; If so, don't deallocate BC yet
   50 54 D0 057B 1137 MOVL R4,R0 ; Set address of BC
   00000000'EF 16 057E 1138 JSB NET$DEALLOCATE ; Deallocate BC
   17 11 0584 1139 BRB 5$ ; and deallocate IOWQE as well
0586 1140 4$:
0586 1141
0586 1142 : If I/O status was not successful, then stop doing any I/O
0586 1143 : on this channel (assume it is in the process of running down).
0586 1144
1D 24 A5 E8 0586 1144 BLBS IOWQE_Q IOSB(R5),10$ ; If I/O failure,
   07 B0 058A 1145 MOVW #EVC$C_NMA_ABS,- ; "Aborted service request"
   1C A5 058C 1146 WQESW EVL CODE(R5)
   01 90 058E 1147 MOVW #EVC$C_NMA_PRSN_ERR,- ; "Receive error"
   1E A5 0590 1148 WQESB EVL DT1(R5)
   FA6B' 30 0592 1149 BSBW NET$EVT_INTRAW ; Log the event record
50 0000'8F 3C 0595 1150 MOVZWL #LEV$C_CIN_DOWN,R0 ; Setup "circuit down" event
   FA63' 30 059A 1151 BSBW SET_DLC_EVT ; Queue event to DLLTRN
   50 55 D0 059D 1152 5$: MOVL R5,R0 ; Get IOWQE address
   00000000'EF 16 05A0 1153 JSB NET$DEALLOCATE ; Deallocate it
   05 05A6 1154 RSB
05A7 1155 10$:
05A7 1156
05A7 1157 : If there is already an unsolicited message received from
05A7 1158 : the remote node waiting for the MOM process to startup,
05A7 1159 : then drop the message on the floor - don't startup a
05A7 1160 : redundant MOM process for the same node.
```



```
51 24 A4 9E 05A7 1161 MOVAB BC_Q_UN SOL_MSGS(R4),R1 ; Get address of unsolicited msg queue
52 51 D0 05AB 1162 MOVL R1,R2 ; Setup for loop
52 62 D0 05AE 1163 15$: MOVL (R2),R2 ; Skip to next msg in list
51 52 D1 05B1 1164 CMPL R2,R1 ; End of list?
OE 13 05B4 1165 BEQL 20$ ; If so, then skip it
06 BB 05B6 1166 PUSHF #M<R1,R2> ; Save registers
OE 29 05B8 1167 CMPC #NIHDR$1Z,- ; Does the NI header match?
38 A2 05BA 1168 IOWQE_G_NIHDR(R2),-
38 A5 05BC 1169 IOWQE_G_NIHDR(R5)
06 BA 05BE 1170 POPR #M<RT,R2> ; Restore registers
EC 12 05C0 1171 BNEQ 15$ ; If it doesn't match, keep looking
06 11 05C2 1172 BRB 30$ ; If match found, drop msg on floor
05C4 1173 20$: ;
05C4 1174 ; Startup a process to deal with the message
05C4 1175 ;
00AE 30 05C4 1176 BSBW STARTUP_MOM ; Start MOM process
05C7 1177 ;
05C7 1178 ; If the process could not be created, re-issue the read
05C7 1179 ; request using the same buffer.
05C7 1180 ;
18 B4 50 E8 05C7 1181 BLBS R0,40$ ; Branch if successful
51 B4 65 OE 05CA 1182 30$: INSQUE (R5),ABC_Q_PND_RCV+4(R4) ; Insert on pending receive queue
51 OE A4 3C 05CE 1183 MOVZWL BC_W_LPD(R4),RT ; Get LPD ID
FEEF 30 05D2 1184 BSBW ISSUE_NI_READ ; Re-issue read request
05 05D5 1185 RSB
05D6 1186 ;
05D6 1187 ; Save PID of MOM process just started in unsolicited message
05D6 1188 ; context block. From now on, this message is "tagged" for
05D6 1189 ; that process: If the process comes in with an ACCESS function,
05D6 1190 ; we give it the message; if the process dies, we deallocate the
05D6 1191 ; message.
05D6 1192 ;
30 A5 51 D0 05D6 1193 40$: MOVL R1,IOWQE_L_PID(R5) ; Save PID of associated MOM process
05DA 1194 ;
05DA 1195 ; Insert the message on the queue waiting for the process to
05DA 1196 ; get started.
05DA 1197 ;
28 B4 65 OE 05DA 1198 INSQUE (R5),ABC_Q_UN SOL_MSGS+4(R4) ; Insert at end of queue
05DE 1199 ;
05DE 1200 ; Re-issue another receive request for this protocol type
05DE 1201 ;
51 05FE 8F 3C 05DE 1202 MOVZWL #IOWQE_C_LENGTH-WQE$C_LENGTH,R1 ; Get additional storage size
50 03 D0 05E3 1203 MOVL #WQE$C_SUB_AST,R0 ; Indicate WQE sub-type
FA17 30 05E6 1204 BSBW WQE$ALLOCATE ; Allocate a WQE - always succeeds
2C A5 B0 05E9 1205 MOVW IOWQE_W_CHAN(R5),- ; Copy channel to datalink
2C A2 05EC 1206 IOWQE_W_CHAN(R2)
34 A5 D0 05EE 1207 MOVL IOWQE_L_BC(R5),- ; Copy backpointer to BC block
34 A2 05F1 1208 IOWQE_L_BC(R2)
12 A5 B0 05F3 1209 MOVW WQE$W_REQIDT(R5),- ; Use the same REQIDT
12 A2 05F6 1210 WQE$W_REQIDT(R2)
18 B4 62 OE 05F8 1211 INSQUE (R2),ABC_Q_PND_RCV+4(R4) ; Insert on pending receive queue
51 OE A4 3C 05FC 1212 MOVZWL BC_W_LPD(R4),RT ; Get LPD ID
FEC1 30 0600 1213 BSBW ISSUE_NI_READ ; Issue another read request
05 0603 1214 90$: RSB
```



```
0604 1216 .SBTTL DLE$MOP_REQUEST - Partner has requested MOP mode
0604 1217 :+
0604 1218 : DLE$MOP_REQUEST - The circuit partner has requested MOP mode
0604 1219 :
0604 1220 : This routine is called when the datalink has received a MOP message
0604 1221 : from the partner node on a point-to-point datalink.
0604 1222 :
0604 1223 : Inputs:
0604 1224 :
0604 1225 : R10/R11 = CRI pointers
0604 1226 : R6 = LPD address
0604 1227 :
0604 1228 : Outputs:
0604 1229 :
0604 1230 : None
0604 1231 :
0604 1232 : R0-R3,R8-R9 are destroyed.
0604 1233 :
0604 1234 : DLE$MOP_REQUEST::
00F0 8F BB 0604 1235 : PUSHR #^M<R4,R5,R6,R7> ; Save registers
02 E2 0608 1236 : BBSS #LPD$V_DLE,- ; Mark circuit in MOP mode
1D 22 A6 060A 1237 : LPD$W_STS(R6),10$ ; If already marked, skip logging event
060D 1238 :
060D 1239 : Log an event indicating the circuit has gone into MOP mode
060D 1240 :
55 00000000'EF 9E 060D 1241 : MOVAB NET$AB_EVT_WQE,R5 ; Get address of common WQE
20 A6 B0 0614 1242 : MOVW LPD$W_PTH(R6),- ; Set LPD ID into WQE
12 A5 0617 1243 : WQESW_REQIDT(R5)
0141 8F B0 0619 1244 : MOVW #EVC$C_DLL_RSC,- ; 'remotely initiated state change'
1C A5 061D 1245 : WQESW_EVL_CODE(R5)
03 90 061F 1246 : MOVW #EVC$C_DLC_POLD_RUNG,- ; Old state = RUNNING
1E A5 0621 1247 : WQESB_EVL_DT1(R5)
04 90 0623 1248 : MOVW #EVC$C_DLC_POLD_MAIN,- ; New state = MAINTAINANCE
1F A5 0625 1249 : WQESB_EVL_DT2(R5)
F9D6' 30 0627 1250 : BSBW NET$EVT_INTRAW ; Log the event record
062A 1251 :
062A 1252 : If circuit is already accessed, then ignore MOP notification
062A 1253 :
2A 22 A6 E0 062A 1254 10$: BBS #LPD$V_ACCESS,- ; Branch if circuit accessed
062C 1255 : LPD$W_STS(R6),40$
062F 1256 :
062F 1257 : If service functions are disabled for this circuit, then
062F 1258 : ignore MOP request, and recycle circuit.
062F 1259 :
24 58 E8 062F 1260 : $GETFLD cri,l,ser ; Get SERVICE flag
063C 1261 : BLBS R8,50$ ; Branch if disabled
063F 1262 :
063F 1263 : Set the circuit substate to 'auto-service'
063F 1264 :
27 06 90 063F 1265 : MOVW #NMASC_LINSS_ASE,- ; Set circuit substate
A6 0641 1266 : LPD$B_SUB_STA(R6)
0643 1267 :
0643 1268 : Startup a process to deal with the message
0643 1269 :
002F 30 0643 1270 : BSBW STARTUP_MOM ; Start MOM process
1A 50 E9 0646 1271 : BLBC R0,50$ ; Branch if unsuccessful
0649 1272 :
```

```

      0649 1273      ; Save PID of MOM process just started in CRI block
      0649 1274      ;
58    51    D0      0649 1275      MOVL    R1,R8      ; Setup PID of created process
      064C 1276      $PUTFLD cri,l,owpid      ; Set DLE owner of circuit
      0659 1277      ;
      0659 1278      ; We can respond to the MOP request. Recycle the circuit
      0659 1279      ; and force it to come up in "MOP" state (because of the
      0659 1280      ; DLE flag).
      0659 1281      ;
50    0000'8F      3C 0659 1282 40$: MOVZWL #LEV$C_LIN_DOWN,R0      ; Setup "line down" event
      F99F'      30 065E 1283      BSBW    SET_DLE_EVT      ; Queue the event
      OD      11 0661 1284      BRB      90$
      0663 1285      ;
      0663 1286      ;
      0663 1287      ; We cannot respond to the MOP request. Recycle the circuit
      0663 1288      ; and force it to come back in regular mode.
      0663 1289      ;
      0663 1290 50$: CLRBIT #LPD$V_DLE,LPD$W_STS(R6) ; Mark circuit in "normal" mode
50    0000'8F      3C 0668 1291      MOVZWL #LEV$C_LIN_DOWN,R0      ; Setup "line down" event
      F990'      30 066D 1292      BSBW    SET_DLE_EVT      ; Queue the event
      00F0 8F      BA 0670 1293 90$: POPR    #^MZR4,R5,R6,R7>      ; Restore registers
      05 0674 1294      RSB
```



```
0675 1296 .SBTTL STARTUP_MOM - Start MOM process
0675 1297 :+
0675 1298 : STARTUP_MOM - Start MOM process for auto-service
0675 1299 : This routine is called to start the MOM process.
0675 1300 :
0675 1301 : Inputs:
0675 1302 :
0675 1303 : R10/R11 = CRI pointers
0675 1304 :
0675 1305 : Outputs:
0675 1306 :
0675 1307 : R0 = Status code
0675 1308 : R1 = IPID of process, if successful
0675 1309 :
0675 1310 : R2-R3,R7-R9 are destroyed.
0675 1311 :
0675 1312 : -
0675 1313 : STARTUP_MOM:
30 BB 0675 1314 : PUSHR #^M<R4,R5> ; Save registers
0677 1315 : $GETFLD cri,s,nam ; Get circuit name
0684 1316 :
0684 1317 : Repeatedly try to startup MOM, and if it fails due to "duplicate
0684 1318 : process name", then try again with another process name until
0684 1319 : it succeeds.
0684 1320 :
59 01 DO 0684 1321 : MOVL #1,R9 ; Start with postfix #1
5E OC C2 0687 1322 10$: : SUBL #12,SP ; Allocate prcnam buffer on stack
5E 5E DD 068A 1323 : PUSHL SP ; Construct descriptor of buffer
OC DD 068C 1324 : PUSHL #12
51 00000005'EF 9E 068E 1325 : MOVAB MOM_PRCNAM,R1 ; Get address of FAO string
50 81 9A 0695 1326 : MOVZBL (R1)+,R0 ; Construct descriptor of FAO string
7E 50 7D 0698 1327 : MOVQ R0,-(SP) ; Push FAO descriptor onto stack
50 5E DO 069B 1328 : MOVL SP,R0 ; Get stack address
069E 1329 : $FAO_S CTRSTR=(R0),- ; Construct process name
069E 1330 : OUTBUF=8(R0),-
069E 1331 : OUTLEN=8(R0),-
069E 1332 : P1=R7,- ; Length of circuit name
069E 1333 : P2=R8,- ; Address of circuit name
069E 1334 : P3=R9 ; Process number
5E 08 CO 06B3 1335 : ADDL #8,SP ; Pop FAO string descriptor
54 8E 7D 06B6 1336 : MOVQ (SP)+,R4 ; R4/R5 = descriptor of process name
52 57 7D 06B9 1337 : MOVQ R7,R2 ; Pass circuit name as SYS$NET
0180 8F BB 06BC 1338 : PUSHR #^M<R7,R8> ; Save circuit name
58 00000000'EF 9E 06C0 1339 : MOVAB MOM_OBJ_NAM,R8 ; Point to ASCII MOM object name
57 88 9A 06C7 1340 : MOVZBL (R8)+,R7 ; Construct descriptor of name
F933' 30 06CA 1341 : BSBW NET$STARTUP_OBJ_NAM ; Startup the object
0180 8F BA 06CD 1342 : POPR #^M<R7,R8> ; Restore circuit name
5E OC CO 06D1 1343 : ADDL #12,SP ; Pop process name buffer
0000'8F 50 B1 06D4 1344 : CMPW R0,#SS$_DUPLNAM ; Process name already exist?
08 12 06D9 1345 : BNEQ 20$ ; If so,
FFA6 59 01 OA F1 06DB 1346 : ACBL #MAX_MOM_PROC,#1,R9,10$ ; Increment number and try again
1A 11 06E1 1347 : BRB 90$ ; Exit with error, but don't log
06E3 1348 : ; any error - too many MOMs already
06E3 1349 20$: :
06E3 1350 : ; If the process could not be created, log an event record.
06E3 1351 :
17 50 E8 06E3 1352 : BLBS R0,90$ ; Branch if successful
```

```
55 00000000 50 DD 06E6 1353 PUSHL R0 ; Save status
      EF 9E 06E8 1354 MOVAB NET$AB_EVT_WQE,R5 ; Get address of common WQE
      07 B0 06EF 1355 MOVW #EVCSC-NMA-ABS,- ; "aborted service request"
      1C A5 06F1 1356 WQESW EVL CODE(R5)
      04 90 06F3 1357 MOVW #EVCSC-NMA-PRSN_LOE,- ; Reason = "line open error"
      1E A5 06F5 1358 WQESB EVL DT1(R5)
      F906' 30 06F7 1359 BSBW NET$EVT_INTRAW ; Log the event record
      50 8ED0 06FA 1360 POPL R0 ; Restore status
      30 BA 06FD 1361 90$: POPR #^M<R4,R5> ; Restore registers
      05 06FF 1362 RSB
```



```
.SBTTL ATTACH_UNSQL_MSG - Attach unsolicited message
0700 1364
0700 1365 :+
0700 1366 : ATTACH_UNSQL_MSG - Attach unsolicited message to newly accessed DWB
0700 1367 :
0700 1368 : This routine is called to search the unsolicited message queue, and
0700 1369 : if one is found for this DLE user, to insert the message onto it's
0700 1370 : private receive queue.
0700 1371 :
0700 1372 : Inputs:
0700 1373 :
0700 1374 : R3 = IOS_ACCESS IRP address
0700 1375 : R4 = DWB address
0700 1376 :
0700 1377 : Outputs:
0700 1378 :
0700 1379 : IRP$L_EXTEND(R3) = Address of CXB containing unsolicited message
0700 1380 : (or zero if no message found)
0700 1381 :
0700 1382 : CXB$W_LENGTH = Message length in bytes (not incl. NI header)
0700 1383 : CXB$C_HEADER = 14-byte NI datalink header
0700 1384 : CXB$C_HEADER+14 = Message
0700 1385 :
0700 1386 : R0-R1 are destroyed.
0700 1387 :-
0700 1388 ATTACH_UNSQL_MSG:
54 55 DD 0700 1389 PUSH R5 ; Save registers
A3 D4 0702 1390 CLRL IRP$L_EXTEND(R3) ; Preset no CXB address
0705 1391 :
0705 1392 : Locate the BC block associated with this circuit.
0705 1393 :
51 00000008'EF 9E 0705 1394 MOVAB BC_QUEUE,R1 ; Get address of BC queue
55 51 D0 070C 1395 MOVL R1,R5 ; Setup for loop
55 65 D0 070F 1396 5$: MOVL (R5),R5 ; Skip to next block in queue
51 55 D1 0712 1397 CMPL R5,R1 ; End of list?
OE 5A 13 0715 1398 BEQL 90$ ; If not found, skip it
3E A5 B1 0717 1399 CMPW BC_W_LPD(R5),- ; Does the LPD ID match?
F1 12 071A 1400 DWB$W_PATH(R4)
071C 1401 BNEQ 5$ ; If not, keep looking
071E 1402 :
071E 1403 : Search for unsolicited message which was "tagged" for
071E 1404 : this process.
071E 1405 :
51 24 A5 9E 071E 1406 MOVAB BC_Q_UNSQL_MSGS(R5),R1 ; Get address of unsolicited msg queue
55 51 D0 0722 1407 MOVL R1,R5 ; Setup for loop
55 65 D0 0725 1408 10$: MOVL (R5),R5 ; Skip to next msg in list
51 55 D1 0728 1409 CMPL R5,R1 ; End of list?
30 44 13 072B 1410 BEQL 90$ ; If so, then skip it
OC A5 D1 072D 1411 CMPL IOWQE_L_PID(R5),- ; Does the IPID match?
F1 12 0730 1412 IRP$L_PID(R3)
0732 1413 BNEQ 10$ ; If not, keep looking
0734 1414 :
0734 1415 : Allocate a CXB from non-paged pool, store the message into
0734 1416 : the block, and insert it into the DWB receive queue.
0734 1417 :
51 51 26 A5 3C 0734 1418 MOVZWL IOWQE_W_MSGLEN(R5),R1 ; Get size of message
0000005A 8F C0 0738 1419 ADDL #CXB$C_OVERHEAD+NIHDRSIZ,R1 ; Compute size of CXB
00000000'EF 16 073F 1420 JSB NET$ALONPAGED ; Allocate from non-paged pool
```

```
08 29 50 E9 0745 1421 BLBC R0,90$ ; If insufficient memory, skip it
0A A2 51 B0 0748 1422 MOVW R1,CXB$W SIZE(R2) ; Set size of structure
62 A2 1B 90 074C 1423 MOVB #DYN$C_CXB,CXB$B_TYPE(R2) ; Set type of structure
56 A2 9E 0750 1424 MOVAB CXB$C_HEADER+ ; Set data area address in CXB
0754 1425 NIHDR$IZ(R2),(R2)
26 A5 B0 0754 1426 MOVW IOWQE_W_MSGLEN(R5),- ; Save message size in CXB
0C A2 0757 1427 CXB$W_LENGTH(R2)
54 A3 52 D0 0759 1428 MOVL R2,IRP$L_EXTEND(R3) ; Save address of CXB
3C BB 075D 1429 PUSHR #^M<R2,R3,R4,R5> ; Save registers
0E 28 075F 1430 MOVC #NIHDR$IZ,- ; Copy NI datalink header
38 A5 0761 1431 IOWQE_G_NIHDR(R5),-
48 A2 0763 1432 CXB$C_HEADER(R2)
55 0C AE D0 0765 1433 MOVL 3*4(SP),R5 ; Recover IOWQE address
26 A5 28 0769 1434 MOVC IOWQE_W_MSGLEN(R5),- ; Copy message
46 A5 076C 1435 IOWQE_G_MSG(R5),-
63 076E 1436 (R3)
3C BA 076F 1437 POPR #^M<R2,R3,R4,R5> ; Restore registers
55 8ED0 0771 1438 90$: POPL R5 ; Restore registers
05 0774 1439 RSB
```



```
0775 1441 .SBTTL DLE$PRC_EXIT - Handle MOM process termination
0775 1442 ;+
0775 1443 ; DLE$PRC_EXIT - Handle MOM process termination
0775 1444 ;
0775 1445 ; This routine is called whenever any process "owned" by NETACP terminates.
0775 1446 ; We must check if we have any unsolicited MOP messages intended for the
0775 1447 ; terminated process, and if so, clean them up.
0775 1448 ;
0775 1449 ; Inputs:
0775 1450 ;
0775 1451 ; R8 = IPID of terminated process
0775 1452 ;
0775 1453 ; Outputs:
0775 1454 ;
0775 1455 ; None
0775 1456 ;
0775 1457 DLE$PRC_EXIT::
0775 1458 ;
0775 1459 ; Scan all broadcast circuits
0775 1460 ;
51 00000008'EF 9E 0775 1461 MOVAB BC_QUEUE,R1 ; Get address of BC queue
55 51 D0 077C 1462 MOVL R1,R5 ; Setup for loop
55 65 D0 077F 1463 5$: MOVL (R5),R5 ; Skip to next block in queue
51 55 D1 0782 1464 CMPL R5,R1 ; End of list?
25 13 0785 1465 BEQL 20$ ; If not found, skip it
0787 1466 ;
0787 1467 ; Deallocate any messages which are intended for this process
0787 1468 ;
52 24 A5 9E 0787 1469 MOVAB BC_Q_UN SOL_MSGS(R5),R2 ; Get address of unsolicited msg queue
53 52 D0 078B 1470 MOVL R2,R3 ; Setup for loop
53 63 D0 078E 1471 10$: MOVL (R3),R3 ; Skip to next msg in list
52 53 D1 0791 1472 15$: CMPL R3,R2 ; End of list?
E9 13 0794 1473 BEQL 5$ ; If so, then continue to next circuit
58 30 A3 D1 0796 1474 CMPL IOWQE_L_PID(R3),R8 ; Does the IPID match?
F2 12 079A 1475 BNEQ 10$ ; If not, keep looking
63 DD 079C 1476 PUSHL (R3) ; Save pointer to next block in list
50 63 OF 079E 1477 REMQUE (R3),R0 ; Remove it from the queue
00000000'EF 16 07A1 1478 JSB NET$DEALLOCATE ; Deallocate the block
53 8ED0 07A7 1479 POPL R3 ; Set R3 to next block in list
E5 11 07AA 1480 BRB 15$ ; Keep looking for more
07AC 1481 20$: ;
07AC 1482 ; If any circuits are in MOP state waiting for the MOM
07AC 1483 ; process to issue its initial ACCESS, then reset them
07AC 1484 ; back into normal state. We recognize this condition
07AC 1485 ; if the OWPID field is still set to the PID, meaning
07AC 1486 ; that the process must never have accessed the DLE
07AC 1487 ; channel (or else we would have cleared it on DEACCESS).
07AC 1488 ;
5B 00000000'EF D0 07AC 1489 MOVL NET$GL_CNR_CRI,R11 ; Point to CRI database
5A D4 07B3 1490 CLRL R10 ; Start at beginning
13 50 E9 07B5 1491 25$: $SEARCH egl_cri,l,owpid ; Search for circuits
F836' 30 07C4 1492 BLBC R0,30$ ; Branch if none found
E8 50 E9 07C7 1493 BSBW NET$LOCATE_LPD ; Locate associated LPD
F828' 30 07CA 1494 BLBC R0,25$ ; If error detected, skip it
50 0000'8F 3C 07CD 1495 BSBW LEAVE MOP STATE ; Return circuit to normal mode
F828' 30 07D0 1496 MOVZWL #LEV$C_LIN_DOWN,R0 ; Setup "line down" event
07D5 1497 BSBW SET_DLC_EVT ; Queue the event
```


NETDLE
V04-000

- NETACP DLE processing
DLE\$PRC_EXIT - Handle MOM process termin

C 2

16-SEP-1984 01:24:27 VAX/VMS Macro V04-00
5-SEP-1984 02:19:17 [NETACP.SRC]NETDLE.MAR;1

Page 37
(19)

DB 11 07DB 1498 BRB 25\$; Keep looping
05 07DA 1499 30\$: RSB
07DB 1500
07DB 1501
07DB 1502 .END

NETDLE
Symbol table

- NETACP DLE processing

D 2

16-SEP-1984 01:24:27 VAX/VMS Macro V04-00
5-SEP-1984 02:19:17 [NETACP.SRC]NETDLE.MAR;1

Page 38
(19)

```

$ST1      = 00000000
$ST2      = 00000006
ABD$C_FIB = 00000001
ABD$C_LENGTH = 00000008
ABD$C_NAME = 00000002
ABD$W_COUNT = 00000002
ABD$W_TEXT = 00000000
ACP$C_STA_F = 00000004
ACP$C_STA_H = 00000005
ACP$C_STA_I = 00000000
ACP$C_STA_N = 00000001
ACP$C_STA_R = 00000002
ACP$C_STA_S = 00000003
ATTACH_UN$OL_MSG = 00000700 R 04
BC_ACCESS = 000001C0 R 04
BC_B_FLAGS = 0000000B G
BC_B_REFCNT = 0000000C G
BC_B_TYPE = 0000000A G
BC_C_LENGTH = 0000002C G
BC_L_FLINK = 00000000 G
BC_M_DELETE = 00000001 G
BC_QUEUE = 00000008 R 02
BC_Q_CUR_RCV = 0000001C G
BC_Q_PND_RCV = 00000014 G
BC_Q_UN$OL_MSGS = 00000024 G
BC_V_DELETE = 00000000 G
BC_W_LD_CHAN = 00000010 G
BC_W_LPD = 0000000E G
BC_W_LP_CHAN = 00000012 G
BC_W_SIZE = 00000008 G
BIT... = 00000001
CCB$C_UCB = 00000000
CNF$C_C_R_FIELD = ***** X 04
CNF$C_G_T_FIELD = ***** X 04
CNF$C_K_EY_SEARCH = ***** X 04
CNF$C_P_UT_FIELD = ***** X 04
CNF$C_A_VANCE = 00000000
CNF$C_QUIT = 00000002
CNF$C_TAKE_CURR = 00000003
CNF$C_TAKE_PREV = 00000001
CXB$B_TYPE = 0000000A
CXB$C_HEADER = 00000048
CXB$C_OVERHEAD = 0000004C
CXB$W_LENGTH = 0000000C
CXB$W_SIZE = 00000008
DDT$C_UN$OL_INT = 00000004
DLE$ACCESS = 00000075 R 04
DLE$BC_DOWN = 000003D2 RG 04
DLE$BC_UP = 00000339 RG 04
DLE$CANCEL = 00000331 R 04
DLE$DEACCESS = 000002A7 R 04
DLE$DISPATCH = 00000000 RG 04
DLE$LPD_STATUS = 00000157 RG 04
DLE$MOP_REQUEST = 00000604 RG 04
DLE$PRC_EXIT = 00000775 RG 04
DLE$SETMODE = 0000021F R 04
DLE_ACC = 00000000 R 02

```

```

DWB$B_SUBSTA = 00000046
DWB$C_DLL_UCB = 00000048
DWB$V_BC = 00000003
DWB$W_DLL_CHAN = 0000004C
DWB$W_FLAGS = 0000000E
DWB$W_PATH = 0000003E
DYN$C_CXB = 0000001B
EVC$C_DLL_LSC = 00000140
EVC$C_DLL_POLD_MAIN = 00000004
EVC$C_DLL_POLD_RUNG = 00000003
EVC$C_DLL_RSC = 00000141
EVC$C_NMA_ABS = 00000007
EVC$C_NMA_PRSN_ERR = 00000001
EVC$C_NMA_PRSN_LOE = 00000004
EXESINSIOQ = ***** X 04
INIT_UN$OL_CHAN = 00000442 R 04
IOSM_CTRL = 00000200
IOSM_STARTUP = 00000040
IOS_ACCESS = 00000032
IOS_ACPCONTROL = 00000038
IOS_DEACCESS = 00000034
IOS_READVBLK = 00000031
IOS_SETMODE = 00000023
IOC$VERIFYCHAN = ***** X 04
IOSB = 00000010 R 02
IOWQE_C_LENGTH = 00000622 G
IOWQE_G_MSG = 00000046 G
IOWQE_G_NIHDR = 00000038 G
IOWQE_L_BC = 00000034 G
IOWQE_L_PID = 00000030 G
IOWQE_Q_IOSB = 00000024 G
IOWQE_W_CHAN = 0000002C G
IOWQE_W_MSGLEN = 00000026
IRP$C_DIAGBUF = 0000004C
IRP$C_EXTEND = 00000054
IRP$C_IOST1 = 00000038
IRP$C_IOST2 = 0000003C
IRP$C_PID = 0000000C
IRP$C_SVAPTE = 0000002C
IRP$C_UCB = 0000001C
IRP$C_WIND = 00000018
IRP$C_FCODE = 00000006
IRP$V_COMPLX = 00000003
IRP$V_FCODE = 00000000
IRP$W_FUNC = 00000020
IRP$W_STS = 0000002A
ISSUE_NI_READ = 000004C4 R 04
LD_PARAMS = 00000011 R 03
LD_SETMODE = 00000053 R 03
LEAVE_MOP_STATE = 000002F8 R 04
LEV$C_DLE_ACC = ***** X 04
LEV$C_LIN_DOWN = ***** X 04
LPD$B_SUB_STA = 00000027
LPD$C_UCB = 00000010
LPD$V_ACCESS = 00000003
LPD$V_BC = 0000000A
LPD$V_DLE = 00000002

```

NE
VO

NETDLE
Symbol table

- NETACP DLE processing

E 2

16-SEP-1984 01:24:27 VAX/VMS Macro V04-00
5-SEP-1984 02:19:17 [NETACP.SRC]NETDLE.MAR;1Page 39
(19)

LPDSV_RUN	= 00000004		
LPDSV_X25	= 00000007		
LPDSW_CHAN	= 00000014		
LPDSW_PTH	= 00000020		
LPDSW_STS	= 00000022		
LP_PARAMS	0000005B	R	03
LP_SETMODE	0000009D	R	03
MAX_MOM_PROC	= 0000000A		
MOM_OBJ_NAM	00000000	R	03
MOM_PRCNAM	00000005	R	03
NET\$AB_EVT_WQE	*****	X	04
NET\$ALLOCATE	*****	X	04
NET\$ALONPAGED	*****	X	04
NETSC_ACT_TIMER	= 0000001E		
NETSC_EFN_ASYN	= 00000002		
NETSC_EFN_WAIT	= 00000001		
NETSC_IPL	= 00000008		
NETSC_MAXACCFD	= 00000027		
NETSC_MAXLINNAM	= 0000000F		
NETSC_MAXLNK	= 000003FF		
NETSC_MAXNODNAM	= 00000006		
NETSC_MAXOBJNAM	= 0000000C		
NETSC_MAX_AREAS	= 0000003F		
NETSC_MAX_LINES	= 00000040		
NETSC_MAX_NCB	= 0000006E		
NETSC_MAX_NODES	= 000003FF		
NETSC_MAX_OBJ	= 000000FF		
NETSC_MAX_WQE	= 00000014		
NETSC_MINBUFSIZ	= 000000C0		
NETSC_TID_ACT	= 00000003		
NETSC_TID_RUS	= 00000001		
NETSC_TID_XRT	= 00000002		
NETSC_TRCTL_CEL	= 00000002		
NETSC_TRCTL_OVR	= 00000005		
NETSC_UTLBUFSIZ	= 00001000		
NET\$DEALLOCATE	*****	X	04
NET\$EVT_INTRAW	*****	X	04
NET\$FIND_LPD	*****	X	04
NET\$GET_CPD_CRI	*****	X	04
NET\$GL_CNR_CRI	*****	X	04
NET\$GL_DLE_UCB	*****	X	04
NET\$LOCATE_LPD	*****	X	04
NET\$M_MAXLNKMSK	= 000003FF		
NET\$STARTUP_OBJ_NAM	*****	X	04
NFBSC_CRI_NAM	= 04020041		
NFBSC_CRI_OWPID	= 04010010		
NFBSC_CRI_SER	= 04000002		
NFBSC_CRI_STA	= 04010013		
NFBSC_CRI_VMSNAM	= 04020042		
NFBSC_OP_EQL	= 00000000		
NIHDRSIZ	= 0000000E		
NMASC_ACC_SHR	= 00000001		
NMASC_LINMC_SET	= 00000001		
NMASC_LINSS_ASE	= 00000006		
NMASC_LINSS_SYN	= 0000000A		
NMASC_PCLI_ACC	= 00000B1E		
NMASC_PCLI_BFN	= 00000451		

NMASC_PCLI_BUS	= 00000AF1		
NMASC_PCLI_CRC	= 00000B1C		
NMASC_PCLI_DCH	= 00000B1B		
NMASC_PCLI_MCA	= 00000B0F		
NMASC_PCLI_MLT	= 00000B19		
NMASC_PCLI_PAD	= 00000B1A		
NMASC_PCLI_PRM	= 00000B18		
NMASC_PCLI_PTY	= 00000B0E		
NMASC_STATE_OFF	= 00000001		
NMASC_STATE_ON	= 00000000		
NSPSC_EXT_LNK	= 0000001E		
NSPSC_MAXHDR	= 00000009		
RCV_DCE_MSG	00000560	R	04
RCV_DLE_MSG_AST	0000054E	R	04
SET_DLL_EVT	*****	X	04
SIZ...	= 00000001		
SS\$DEVALLOC	*****	X	04
SS\$DEVINACT	*****	X	04
SS\$DUPLNAM	*****	X	04
SS\$FILNOTACC	*****	X	04
SS\$ILLIOFUNC	*****	X	04
SS\$IVMODE	*****	X	04
SS\$NORMAL	*****	X	04
SS\$NOSUCHDEV	*****	X	04
STARTUP_MOM	00000675	R	04
SYSSASSIGN	*****	GX	04
SYSSDASSGN	*****	GX	04
SYSSFAO	*****	X	04
SYSSQIO	*****	GX	04
SYSSQIOW	*****	GX	04
TID_C_READSUP	= 00000001		
TRSC_MAXHDR	= 0000001C		
TRSC_NI_ALLEND1	= 040000AB		
TRSC_NI_ALLEND2	= 00000000		
TRSC_NI_ALLROU1	= 030000AB		
TRSC_NI_ALLROU2	= 00000000		
TRSC_NI_PREFIX	= 000400AA		
TRSC_NI_PROT	= 00000360		
TRSC_PRI_ECL	= 0000001F		
TRSC_PRI_RTHRU	= 0000001F		
UCB\$C_DDT	= 00000088		
WQES\$ALLOCATE	*****	X	04
WQESB_EVL_DT1	= 0000001E		
WQESB_EVL_DT2	= 0000001F		
WQESC_LENGTH	= 00000024		
WQESC_QUAL_DLE	= 00000004		
WQESC_SUB_AST	= 00000003		
WQES\$DEALLOCATE	*****	X	04
WQES\$INSQUE	*****	X	04
WQESL_ACTION	= 0000000C		
WQES\$RESET_TIM	*****	X	04
WQESW_EVL_CODE	= 0000001C		
WQESW_REQIDT	= 00000012		
\$\$	= 00000000		

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000622 (1570.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
NET_IMPURE	00000018 (24.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
NET_PURE	000000A5 (165.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
NET_CODE	000007DB (2011.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	27	00:00:00.11	00:00:00.57
Command processing	152	00:00:01.10	00:00:04.42
Pass 1	831	00:00:32.24	00:00:43.49
Symbol table sort	0	00:00:04.69	00:00:05.06
Pass 2	376	00:00:06.40	00:00:08.42
Symbol table output	31	00:00:00.21	00:00:00.22
Psect synopsis output	4	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1423	00:00:44.80	00:01:02.23

The working set limit was 2000 pages.
178619 bytes (349 pages) of virtual memory were used to buffer the intermediate code.
There were 180 pages of symbol table space allocated to hold 3209 non-local and 73 local symbols.
1502 source lines were read in Pass 1, producing 26 object records in Pass 2.
58 pages of virtual memory were used to define 53 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[SHRLIB]NMALIBRY.MLB;1	1
_\$255\$DUA28:[SHRLIB]EVCDEF.MLB;1	1
_\$255\$DUA28:[NETACP.OBJ]NETDRV.MLB;1	0
_\$255\$DUA28:[NETACP.OBJ]NET.MLB;1	13
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	10
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	17
TOTALS (all libraries)	42

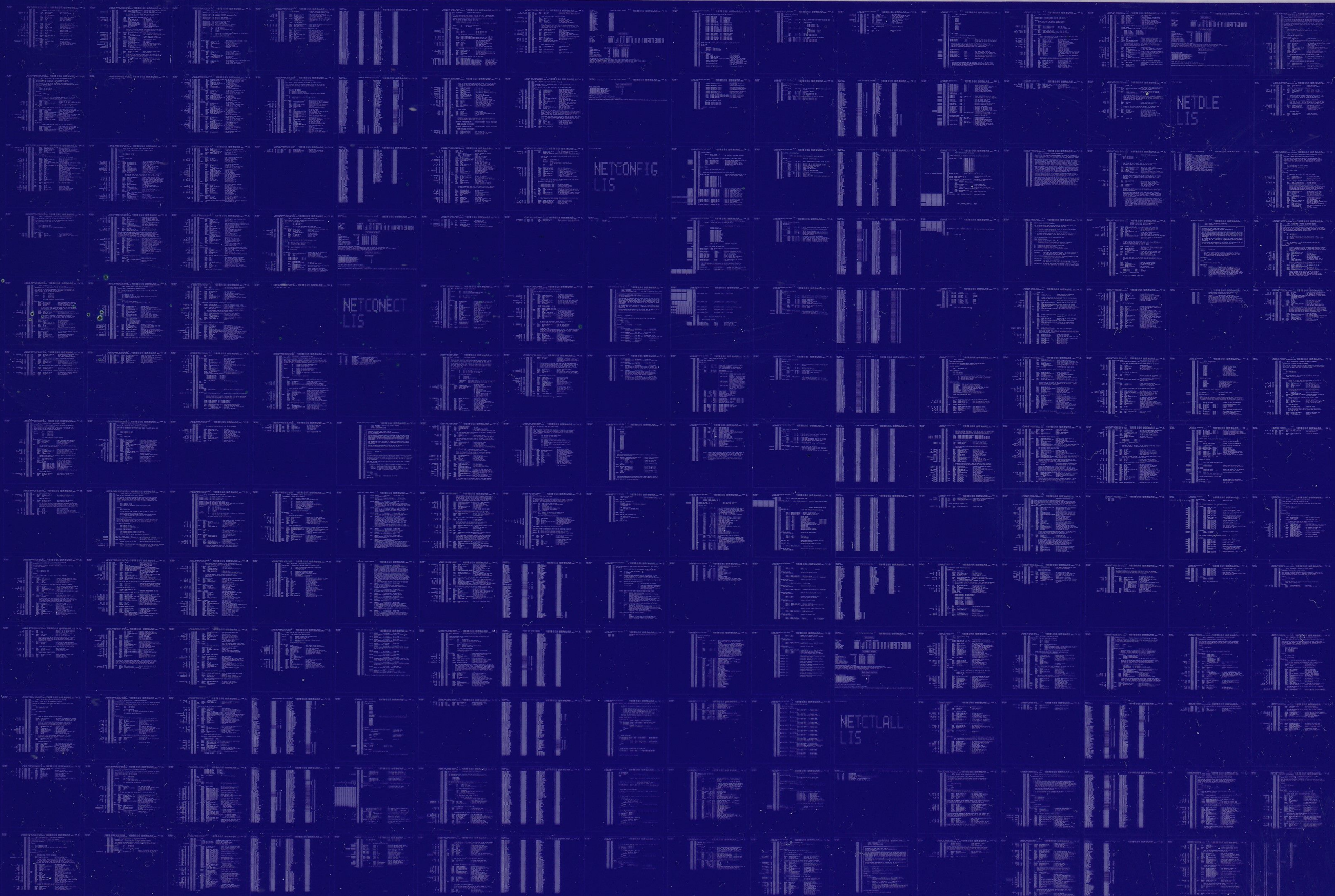
3530 GETS were required to define 42 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:NETDLE/OBJ=OBJ\$:NETDLE MSRC\$:NETDLE/UPDATE=(ENH\$:NETDLE)+EXECML\$/LIB+LIB\$:NET/LIB+LIB\$:NETDRV/LIB+SHRLIB\$:EVCDEF/LIB+

0275 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0276

AH-BT13A-SE
 VAX/VMS V4.

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY